

# **ECHO 9.0 Client Partner's Guide**

For ECHO Version 9.0

NASA

April 25, 2007

1.0 Edition

**This page is intentionally left blank.**

# Table of Contents

|  |      |
|--|------|
| Table of Contents .....  | i    |
| Table of Figures .....   | vii  |
| Table of Code Listings .....                                       | ix   |
| Preface .....  | xiii |
| Conventions .....  | xiii |
| Chapter 1: Before You Begin .....                                  | 1    |
| Tasks That You Will Perform as a Client Partner .....              | 2    |
| Skills You Will Need as a Client Partner .....                     | 3    |
| ECHO Concept and Design .....                                      | 3    |
| ECHO as a Spatially Enabled Metadata Search and Order System ..... | 4    |
| Security .....   | 5    |
| Supported Platforms .....  | 5    |
| ECHO as a Service Registry .....                                   | 6    |
| ECHO Capability and Functionality .....                            | 7    |
| ECHO 9.0 Benefits to Client Partners .....                         | 8    |
| Ease of Participation .....  | 8    |
| Cost to Field .....  | 8    |
| Cost to Operate .....  | 8    |
| Extensibility .....  | 8    |
| Chapter 2: The Basics .....  | 9    |
| Locating the ECHO Web Services .....                               | 9    |
| ECHO Error Handling .....  | 11   |
| ECHO Globally Unique Identifiers (GUIDS) .....                     | 11   |
| ECHO Entities .....  | 11   |
| Users .....  | 11   |
| Roles .....  | 12   |
| Queries .....  | 12   |
| Results .....  | 13   |
| Catalog Items .....  | 13   |
| Orders .....   | 13   |

|   |    |
|---|----|
| Groups.....   | 15 |
| Extended Services.....                                      | 15 |
| Chapter 3: Logging In, Setting Up, and Getting Started..... | 17 |
| Creating and Managing ECHO Sessions .....                   | 17 |
| Logging In to ECHO.....                                     | 17 |
| Logging Out of ECHO.....                                    | 18 |
| Interacting with ECHO .....                                 | 19 |
| User Accounts.....  | 19 |
| Creating a User Account.....                                | 20 |
| Chapter 4: Querying for Earth Science Metadata.....         | 23 |
| Formatting Your Query Results.....                          | 25 |
| Saving Complex Queries .....                                | 28 |
| Handling Large Result Sets .....                            | 29 |
| Saving Results.....   | 31 |
| Visibility of Results .....                                 | 31 |
| Restricted Items .....                                      | 32 |
| Deleted Items .....   | 32 |
| Querying for Orderable Data .....                           | 32 |
| Sample Queries .....  | 32 |
| Using Subscriptions to Automate Queries.....                | 36 |
| Creating a Subscription.....                                | 36 |
| Deleting a Subscription.....                                | 40 |
| Listing Subscriptions .....                                 | 40 |
| Chapter 5: The ECHO Alternative Query Language.....         | 47 |
| Discovery Search .....                                      | 50 |
| Inventory Search .....                                      | 52 |
| Common Search Elements.....                                 | 54 |
| Case Insensitive Searching .....                            | 54 |
| CampaignShortName.....                                      | 54 |
| DatasetID .....   | 54 |
| OnlineOnly.....   | 55 |
| ECHO Insert Date .....                                      | 55 |

|   |    |
|---|----|
| ECHO Last Update .....                      | 56 |
| Sensor Name .....                           | 56 |
| Source Name .....                           | 57 |
| Spatial .....                               | 57 |
| Spatial Type: NORMAL .....                  | 60 |
| Spatial Type: GLOBAL .....                  | 62 |
| Spatial Type: ORBIT .....                   | 62 |
| Temporal .....                              | 64 |
| Orderable Items .....                       | 65 |
| Version ID .....                            | 65 |
| Additional Attributes .....                 | 65 |
| Search Elements for Discovery Queries ..... | 67 |
| Parameter .....                             | 67 |
| Processing Level .....                      | 67 |
| Additional Attribute Name .....             | 68 |
| Short Name .....                            | 68 |
| Spatial Keywords .....                      | 69 |
| Temporal Keywords .....                     | 69 |
| Archive Center .....                        | 70 |
| Search Elements for Inventory Queries ..... | 70 |
| Granules with Browse Data .....             | 70 |
| Percentage of Cloud Cover .....             | 70 |
| Day, Night, or Both Granules .....          | 70 |
| Only Global Granules .....                  | 71 |
| ECHO Granule IDs .....                      | 71 |
| Granule UR .....                            | 72 |
| Two-Dimensional Coordinate System .....     | 72 |
| ProducerGranuleID .....                     | 73 |
| PGE Name .....                              | 74 |
| PGE Version .....                           | 74 |
| Collection Short Name .....                 | 75 |
| Measured Parameters .....                   | 75 |

|   |    |
|---|----|
| ProviderInsertDate .....                    | 76 |
| ProviderProductionDate.....                 | 77 |
| Miscellaneous Elements.....                 | 78 |
| List .....                                  | 78 |
| Text Pattern.....                           | 78 |
| Value .....                                 | 79 |
| Date .....                                  | 79 |
| Date Range.....                             | 79 |
| DifEntryId .....                            | 80 |
| Chapter 6: Ordering Data Through ECHO ..... | 81 |
| Order Options.....                          | 83 |
| Creating an Order.....                      | 84 |
| Common Selection.....                       | 84 |
| Adding Order Items .....                    | 85 |
| Updating Order Items .....                  | 86 |
| Removing Order Items.....                   | 86 |
| Removing Orders and Canceling Orders .....  | 86 |
| Setting User Information for an Order.....  | 87 |
| Validating an Order.....                    | 87 |
| Requesting a Quote for an Order .....       | 87 |
| Submitting an Order.....                    | 87 |
| Tracking and Canceling Orders .....         | 88 |
| Order States.....                           | 89 |
| Restricted or Deleted Order Items .....     | 89 |
| Chapter 7: Event Notification Service ..... | 91 |
| Event Subscriptions .....                   | 91 |
| Delivery Modes.....                         | 91 |
| Filtering Events.....                       | 91 |
| Subscription Expiration .....               | 91 |
| Renewing Event Subscriptions .....          | 92 |
| Creating Event Subscriptions.....           | 92 |
| Removing Event Subscriptions.....           | 92 |

|  |     |
|--|-----|
| Chapter 8: Invocation Service.....                                 | 93  |
| Limitations .....  | 93  |
| Primitive Type Arguments.....                                      | 93  |
| String Results .....   | 93  |
| Invocation State .....   | 93  |
| Client Operations .....  | 94  |
| Invoke Operation .....   | 94  |
| Arguments.....   | 94  |
| ECHO Path URI Parameters.....                                      | 94  |
| Returned Invocation GUID.....                                      | 95  |
| Get Results .....  | 95  |
| Service Partner Operations .....                                   | 95  |
| SOAP Header.....   | 95  |
| Invocation Utility Service.....                                    | 96  |
| Invocation Sequence Diagrams.....                                  | 97  |
| Appendix A: ECHO Path URIs .....                                   | 101 |
| Format.....  | 101 |
| Behavior.....  | 101 |
| Appendix B: Java-Specific Information .....                        | 103 |
| Java: Axis Time-out Occurs While Calling Long Running Methods..... | 103 |
| Appendix C: Acronyms Used in ECHO .....                            | 105 |
| Appendix D: ECHO Error Handling.....                               | 107 |
| Appendix E: Results DTDs.....                                      | 109 |
| ECHO Collection Results DTD .....                                  | 109 |
| ECHO Granule Results DTD.....                                      | 115 |
| Appendix F: Best Practices for Queries.....                        | 119 |
| Best Practices for Running Faster Queries .....                    | 119 |
| Efficient Spatial Queries .....                                    | 119 |
| Notes about Queries .....  | 119 |
| Index .....  | 121 |

**This page is intentionally left blank.**



Table of Figures

Figure 1 ECHO Overview ..... 3

Figure 2: Allowed Geometries for Spatial Data and Query Windows ..... 58

Figure 3: Order of Points in a Polygonal Ring ..... 58

Figure 4: Examples of Spatial Regions..... 59

Figure 5: Polygon Area Defined Depends on the Number of Points and the Coordinate System Used..... 61

Figure 6: Example Data Orientations ..... 62

Figure 7: ECHO Order Structure ..... 81

Figure 8: ECHO Invocation Service Sequence..... 98

Figure 9: Asynchronous Invocation Sequence ..... 99

**This page is intentionally left blank.**

## Table of Code Listings

|   |    |
|---|----|
| Code Listing 1: Logging In as a Guest .....   | 17 |
| Code Listing 2: Logging In as a Registered User .....   | 18 |
| Code Listing 3: Logging out of ECHO.....  | 18 |
| Code Listing 4: Getting the ECHO Version Number .....   | 19 |
| Code Listing 5: Creating an ECHO Account.....   | 21 |
| Code Listing 6: Executing a Simple Query .....  | 27 |
| Code Listing 7: Executing a Query for Specific Metadata .....   | 27 |
| Code Listing 8: Saving a Complex Query .....  | 28 |
| Code Listing 9: Executing a Saved Query .....   | 28 |
| Code Listing 10: Getting the Names of Saved Queries .....   | 28 |
| Code Listing 11: Removing Saved Queries.....  | 29 |
| Code Listing 12: Paging Through Query Results .....   | 30 |
| Code Listing 13: Saving a Result Set for Future Use .....   | 31 |
| Code Listing 14: Removing Saved Result Sets .....   | 31 |
| Code Listing 15: Sample Collection Query (Discovery Search) .....   | 32 |
| Code Listing 16: Sample Collection Query from Two Providers (Discovery Search).....                       | 33 |
| Code Listing 17: Sample Collection Query with Temporal and Spatial Constraints<br>(Discovery) .....       | 33 |
| Code Listing 18: Sample Collection Query with Complex Temporal and Spatial<br>Conditions (Discovery)..... | 34 |
| Code Listing 19: Sample Collection Query Using Provider Specific Attributes<br>(Discovery) .....          | 35 |
| Code Listing 20: Creating a Basic Subscription .....  | 37 |
| Code Listing 21: Removing a Subscription .....  | 40 |
| Code Listing 22: Listing Subscriptions.....   | 40 |
| Code Listing 23: Sample Granule Query Using Various Granule Conditions (Inventory)<br>.....               | 41 |
| Code Listing 24: Sample Granule Query Using Spatial and Temporal Bounds (Inventory)<br>.....              | 42 |
| Code Listing 25: Sample Granule Query Using Sensor Name (Inventory).....                                  | 43 |
| Code Listing 26: Sample Granule Query Using Temporal Constraints (Inventory) .....                        | 44 |

|   |    |
|---|----|
| Code Listing 27: Sample Granule Query Using Additional Provider Specific Attributes (Inventory) .....                     | 45 |
| Code Listing 28: Sample Granule Query Using Additional Provider Specific Attributes with Complex Values (Inventory) ..... | 46 |
| Code Listing 29: General AQL Structure .....  | 47 |
| Code Listing 30: Structure of a Discovery Query with Multiple Collection Conditions .                                     | 48 |
| Code Listing 31: Structure of an Inventory Query with Multiple Granule Conditions ....                                    | 49 |
| Code Listing 32: Using the Negated Attribute on Collection Conditions .....   | 50 |
| Code Listing 33: Spatial Search Element .....   | 60 |
| Code Listing 34: A Multi-Polygon Spatial Query .....  | 63 |
| Code Listing 35: A Single Point Spatial Query .....   | 63 |
| Code Listing 36: A Spatial Query Using a Line Defined by GML Elements .....   | 63 |
| Code Listing 37: Temporal Search Example .....  | 64 |
| Code Listing 38: Temporal Example with Different Day Notations .....  | 64 |
| Code Listing 39: Using the Orderable Condition .....  | 65 |
| Code Listing 40: Version ID in a Collection Condition .....   | 65 |
| Code Listing 41: Version ID in a Granule Condition with a Pattern .....   | 65 |
| Code Listing 42: Additional Attributes Example with an AND Relationship .....   | 66 |
| Code Listing 43: Additional Attribute Example with an OR Relationship .....   | 66 |
| Code Listing 44: Sample Parameter Search .....  | 67 |
| Code Listing 45: Processing Level Search Element .....  | 67 |
| Code Listing 46: Processing Level Element with Text Pattern .....   | 68 |
| Code Listing 47: Additional Attribute Search Element .....  | 68 |
| Code Listing 48: Short Name Search Element .....  | 68 |
| Code Listing 49: Spatial Keywords Search Element .....  | 69 |
| Code Listing 50: Spatial Keywords Element with Text Pattern .....   | 69 |
| Code Listing 51: Temporal Keywords Element .....  | 69 |
| Code Listing 52: Archive Center Search Element .....  | 70 |
| Code Listing 53: Browse-only Search Element .....   | 70 |
| Code Listing 54: Cloud Cover Search Element .....   | 70 |
| Code Listing 55: Search Element for Granules Gathered During Daylight Only .....  | 71 |
| Code Listing 56: Search Element for Granules Gathered During the Nighttime Only ....                                      | 71 |

|  |     |
|--|-----|
| Code Listing 57: Search Element for Granules Gathered During Daylight, Nighttime, or Both..... | 71  |
| Code Listing 58: Global Granules Only Search Element .....                                     | 71  |
| Code Listing 59: Granule ID Search Element .....   | 72  |
| Code Listing 60: Granule UR Search Element.....  | 72  |
| Code Listing 61: TwoDCoordinateSystem Search Element.....                                      | 73  |
| Code Listing 62: TwoDCoordinateSystem Search Element with a Single Value .....                 | 73  |
| Code Listing 63: ProducerGranuleID Search Element.....   | 74  |
| Code Listing 64: PGE Name Search Element .....   | 74  |
| Code Listing 65: PGE Version Search Element .....  | 74  |
| Code Listing 66: Collection Short Name Search Element.....                                     | 75  |
| Code Listing 67: Measured Parameter Search Element .....                                       | 75  |
| Code Listing 68: Measured Parameter Search Element .....                                       | 76  |
| Code Listing 69: Provider Insert Date Search Element .....                                     | 77  |
| Code Listing 70: Provider Production Date Search Element.....                                  | 77  |
| Code Listing 71: The List Element.....   | 78  |
| Code Listing 72: Text Pattern Element.....   | 79  |
| Code Listing 73: Getting Order Options for a Catalog Item .....                                | 84  |
| Code Listing 74: Creating a Simple Order .....   | 85  |
| Code Listing 75: Adding Items to an Existing Order .....                                       | 85  |
| Code Listing 76: Updating an Item in an Order .....  | 86  |
| Code Listing 77: Removing an Item from an Order .....  | 86  |
| Code Listing 78: Creating an Event Subscription.....   | 92  |
| Code Listing 79: Invocation Service SOAP Header Example.....                                   | 96  |
| Code Listing 80: Catching Exceptions from ECHO.....  | 108 |

**This page is intentionally left blank.**

## Preface

This document describes the Version 9.0 operational release of the Earth Observing System (EOS) Clearinghouse (ECHO) system.

### Conventions

All references to time are in Universal Time Coordinated (UTC).

Data Partners are also referred to as Data Providers.

Client Partners are also referred to as Client Developers.

Words in **bold** text are key words or concepts.

---

`Programming examples use a fixed width font and this color font.`

---

---

`Comments (denoted by // within examples) use this color font.`

---

*Best practices or warnings appear in italicized, boxed text.*

**This page is intentionally left blank.**



## Chapter 1: Before You Begin

The primary reason for designing the EOS ClearingHOuse (ECHO) was to increase access to Earth science data and services by providing a system with a machine-to-machine interface, that is, an Application Programming Interface (API).






ECHO functions as a metadata clearinghouse of Earth science metadata for a wide variety of partners, enabling the science community to exchange information. Data Partners provide the ECHO community with metadata representing their Earth science data holdings. ECHO technology in turn provides services for Client Partners and Data Partners and supports efficient discovery and access to Earth science data.

ECHO also functions as an order broker for the data, and offers services applied to that data. ECHO provides a portal on the internet where ECHO clients can search the metadata for information they wish to order.

Client applications can access data holdings via order distribution or online access. Data Partners retain complete control over what metadata are represented in ECHO including inserting new metadata, modifying existing metadata and removing old metadata, and controlling access to their metadata.

## Tasks That You Will Perform as a Client Partner

You will usually perform these tasks in the order shown below:

- Logging in and getting started
  - Creating and managing ECHO sessions
  - Creating and managing user accounts Chapter 3
- Querying for Earth Science Data
  - Formatting query results
  - Saving complex queries
  - Handling large result sets
  - Saving results
  - Using subscriptions to automate queries
    - Creating and deleting subscriptions
    - Listing subscriptions Chapter 4 and 5
- Ordering data through ECHO
  - Adding order items
  - Updating order items
  - Removing order items
  - Removing orders
  - Setting user information for an order
  - Validating orders
  - Requesting a quote for an order
  - Submitting an order Chapter 6
- Using Event Notification Service
  - Setting delivery modes
  - Filtering events
  - Creating event subscriptions
  - Removing event subscriptions
  - Renewing event subscriptions Chapter 7
- Using Invocation Service
  - Invoking an operation
  - Getting results Chapter 8

## Skills You Will Need as a Client Partner

Since ECHO uses platform-independent web service definitions for its API, there are no requirements for a client programming language. All examples in this document are in snippets of Java code; however, the code samples provided could be translated to any web service-capable language.

As an ECHO Client Partner, you need to be familiar with basic software development and Service Oriented Architecture (SOA) concepts such as:

- XML and XML Schema (XSD)
- Web Service Definition Language (WSDL)
- Service-based Application Programmer's Interface (API)
- Client/Server-based programming (client stubs, remote endpoints, etc.)

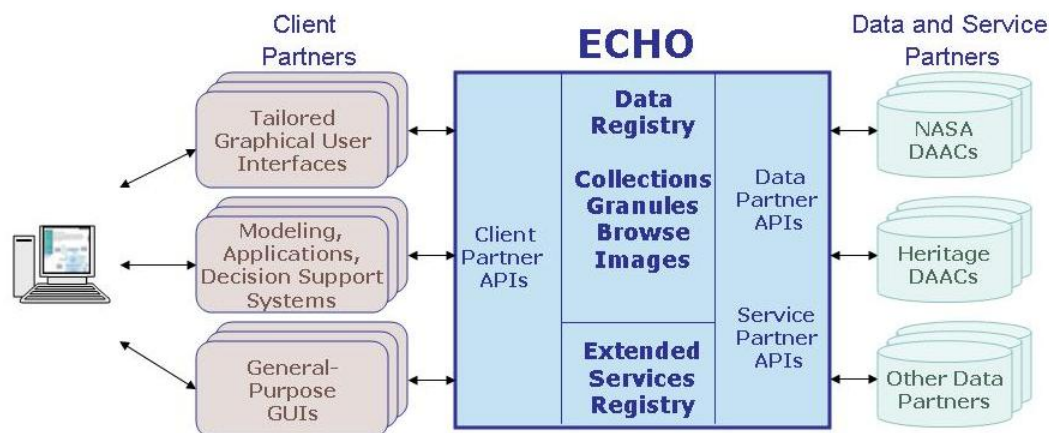
## ECHO Concept and Design

NASA's Earth Science Data and Information System (ESDIS) has built ECHO based on Extensible Markup Language (XML) and Web Service technologies. ECHO interfaces with different clients and users through its series of Application Program Interfaces (APIs). ECHO is an open system with published APIs available to the ECHO Development and User community.

ECHO is a middleware application, and interacting with ECHO means interacting with the ECHO API. There is typically a user-focused client application interacting with ECHO's API on behalf of an end user. This client may be a generic, query and order-based client, or may be specific to an end user's research, mission, or general area of interest. ECHO incorporates a Universal Description, Discovery, and Integration (UDDI) registry to facilitate registration, discovery, and invocation of services related to the ECHO holdings.

Internally, ECHO specifies APIs and provides middleware components, including data and service search and access functions, in a layered architecture. The figure below depicts the ECHO system context in relation to its public APIs.

Figure 1 ECHO Overview



All metadata is stored in an Oracle database with spatial extensions. The metadata model is derived primarily from that used by the Earth Observing System Data and Information System (EOSDIS) Core System (ECS). For more details about the ECHO model, refer to [http://www.echo.nasa.gov/data\\_partners/ECHO1\\_70.shtml](http://www.echo.nasa.gov/data_partners/ECHO1_70.shtml).

Key features of the ECHO architecture are:

- *Ease of Partner Participation* – Designed to be low-cost and minimally intrusive, ECHO offers a set of standard ways for partners to interface with the system and a metadata exchange approach that accommodates existing partners and technology.
- *Data Model Consistency* – To mitigate the risk of being unable to match all possible partner data models, ECHO has prototyped a Metadata Mapping Tool to translate non-standard formats upon ingest into ECHO.
- *Open System/Published APIs* – To accommodate independent ECHO clients, ECHO uses an open system approach and publishes domain APIs. These APIs are independent of the underlying transport protocols used. ECHO communicates using WS-I Basic Profile v1.0 compliant web services. The ECHO 9.0 API is located at <http://www.echo.nasa.gov/reference/reference.shtml>.

Interactions with ECHO may involve user interactions in real time or may be machine to machine.

- *Evolutionary Development* – The ECHO system is being developed incrementally to allow for insight and feedback during the development cycle. Industry trends are followed and the use of commercial, off-the-shelf (COTS) products is optimized.
- *Extensibility of Client User Interfaces and Capabilities* – ECHO extensibility is ensured by its component architecture, which allows new capabilities and functions to be plugged in, modeling relationships between services/APIs/UIs, and continued prototyping. ECHO's current focus is on the middleware and on enabling many different types of user interfaces via its APIs.

### **ECHO as a Spatially Enabled Metadata Search and Order System**

Oracle enables the ECHO system to interact with spatially enabled Earth Science metadata by use of spatial extensions into the system and business logic within the system that understands how to interact with that metadata. In addition, a second ECHO interface (auto-ingest) allows metadata updates to go directly into the database, bypassing the message-passing API. The File Transfer Protocol (FTP) server is configured to receive these update files, which are expressed in XML conforming to two DTDs, one for granules (or inventory) and one for collections (or datasets). The FTP server is located here:

<FTP://ingest.echo.nasa.gov>

- All DTDs are defined here:

<http://www.echo.nasa.gov/reference/reference.shtml>

Oracle's spatial capabilities support queries for ECHO metadata whose spatial extent is described within the system. A Data Partner can define the spatial extent of a granule or a collection with different spatial constructs (for example, point and polygon). A Client Partner can then construct a search using a point, a line, or a polygon (or multiple polygon) spatial type, and ECHO responds with data whose spatial region intersects the described region.

ECHO provides services for interacting with its **Catalog** of metadata. Queries can be performed in a number of ways; result formats can be specified, and the resulting data sets can be incrementally accessed so that large return sets can be handled gracefully. ECHO also supports constructing, submitting, and tracking orders for the data that the metadata represents. ECHO supports both an embedding of a Uniform Resource Locator (URL) within the metadata for accessing the data (which the client simply accesses via Hypertext Transfer Protocol [HTTP]), and a more complicated order process in which quotes and order options are accommodated.

ECHO incorporates the ECS concept of granules and collections and defines separate DTDs for updating each, under the assumption that granules will indicate which collection is considered their "primary" collection. "Primary collection" means the collection that owns the granule, that is, the collection from which that granule can be ordered.

A **collection** is a grouping of granules that all come from the same source. Collections have information that is common across all the granules they contain and a template for describing additional attributes not already part of the metadata model.

A **granule** is the smallest aggregation of data that can be independently managed (described, inventoried, and retrieved). Granules have their own metadata model and support values associated with the additional attributes defined by the owning collection.

A third type of metadata, which is not spatially enabled but useful, is **browse metadata**, which provide a high-level view of granule or collection metadata and cross-referencing to other granules or collections.

## Security

The ECHO system supports Secure Sockets Layer (SSL)-based communication, which a client can use to pass passwords or other sensitive information securely. Internally, the systems are fire walled to prevent unintended access.

## Supported Platforms

The ECHO system supports clients capable of initiating an HTTP connection from a variety of programming languages.

## ECHO as a Service Registry

The ECHO Extended Services Registry component is a public UDDI registry deployment. A UDDI registry enables organizations offering services to register and classify those services so that service consumers may discover them. For Web Services, a UDDI registry stores and provides information sufficient for service consumers to find the services offered by the partner and provides a central location for information publication and discovery. However, it does not participate directly in operations that may occur between consumers and partners.

There are standard programming interfaces (APIs) defined for all instances of UDDI registries. The standard UDDI interface comprises two distinct interfaces: the "Inquiry API" and the "Publishing API". The ECHO system makes the Inquiry API, which is used for discovering and retrieving information about registry content, available as a public interface. This interface describes a web service, which receives SOAP-formatted messages. In accordance with the UDDI specification, these messages "all behave synchronously and are required to be exposed via HTTPPOST only." You can find information about the UDDI Inquiry API at the following location:

<http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv3>

## ECHO Capability and Functionality

ECHO provides an infrastructure that allows various communities to share tools, services and metadata. As a metadata clearinghouse, it supports many data access paradigms such as navigation and discovery. As an order broker, ECHO forwards orders for data discovered through the metadata query process to the appropriate Data Partners for order fulfillment. As a service broker, ECHO decentralizes end user functionality and supports interoperability of distributed functions.

Although this Guide focuses on the needs of Data Partners, ECHO supports the following different, nonexclusive types of Partners:

- *Data Partners* – Organizations that supply metadata representing their data holdings to the ECHO database.
- *Client Partners* – Organizations that participate by developing software applications to access the Earth Science metadata in the ECHO database.
- *Service Partners* – Organizations that participate by advertising their Earth Science-related services to the user community via ECHO, which maintains service descriptions in a Service Catalog (either special services, or services that are available as an option on a selected set of granules/collections) and supports the user in ordering those services.

ECHO addresses science user needs through a set of well-defined and open interfaces upon which the user community can build its own client applications. In this way, ECHO supports extendable, flexible user interfaces, allowing industry and the science community to drive the progress of available Earth Science applications. For more complete information about client applications, refer to the companion piece to this Guide, the *ECHO 9.0 Data Partner's Guide*.

The ECHO approach allows users to build their own user interfaces to ECHO, rather than being limited to the data search and order system provided by NASA. For Data Partners, ECHO offloads the burden of providing the system resources required for searching and gives users the flexibility to support community-specific services and functionality. ECHO's interoperability features allow all participants to benefit from the distributed development of functions, again reducing dependence on NASA resources.

## ECHO 9.0 Benefits to Client Partners

To address the system vision outlined above, ECHO has responded to a set of system drivers, that is, reasons for upgrading. These drivers, derived from functional, organizational and operational concerns expressed by the user community, determined the architectural approach and the types of technical solutions used in building the ECHO 9.0 system.

### Ease of Participation

The primary goal of ECHO is to enable organizations to participate in making their resources and capabilities available to the Earth Science community. To facilitate participation by these organizations, ECHO has:

- Minimized the number of requirements that a partner must meet to participate.
- Involved partners in the system's development cycle and requirements definition.
- Selected metadata insert and update mechanisms based on current standard industry practice (for example, XML) that most databases can generate automatically.
- Provided mapping capabilities to convert from one XML representation into another.

### Cost to Field

While aggressive in the capabilities it is targeted to support, ECHO minimizes the Cost to Field by continually evaluating performance and functionality against costs, for example, licensing of Commercial Off-the-Shelf (COTS) applications, amount of custom code required, hardware platform requirements, and complexity of networking and installation.

### Cost to Operate

Once fielded, ECHO seeks to minimize the cost to operate the system by making it easier to use, thereby minimizing the load on operations staff.

### Extensibility

ECHO is being built with long-term extensibility foremost in mind. To enable emerging techniques and strategies for Earth Science research, ECHO has:

- Adopted a “design for change” as a goal at the beginning of ECHO development.
- Built in the capability to limit the impact of changes to the API on the configuration file, the service interface, and the business logic that implement the function.
- Developed test tools to regression test large portions of functionality automatically overnight, so that when changes are made, undesirable impacts can be discovered quickly.
- Adopted a layered ECHO architecture to allow changes to one component of the system without affecting other components.



## Chapter 2: The Basics

One of ECHO's goals is to increase access to Earth Science data and services by providing a system with a machine-to-machine interface. For that reason, ECHO provides an Application Programming Interface (API). This section describes how to communicate with ECHO using its Web Services API.

### Locating the ECHO Web Services

To access a particular service through the ECHO Web Services API, refer to the ECHO website.

The table below shows each ECHO service and a brief description of its capabilities. The ECHO 9.0 Web Services API documentation describes in detail the services along with their operations and parameters, and is currently available online at: <http://api.echo.nasa.gov/echo/ws/v9/index.html>.

To access the Web Service Description Language (WSDL) document that describes a service, attach the suffix .wsdl from the API page following this format: <http://api.echo.nasa.gov/echo-wsdl/v9/<Service Endpoint>.wsdl>.

**Table 1: Description of ECHO Web Services**

| Service Name       | Description   | Service Endpoint          |
|--------------------|---|---------------------------|
| Administration     | ECHO Operations housekeeping functions  | /AdministrationService    |
| Authentication     | Core user authentication capabilities—provides session management token services  | /AuthenticationService    |
| Catalog            | Data warehouse searching and exploration  | /CatalogService           |
| Data Management    | Data Partner service to support ECHO cataloged data                               | /DataManagementService    |
| Event Notification | User service to manage subscriptions to receive notifications of events from ECHO | /EventNotificationService |
| Extended Services  | Registration, classification and maintenance of Earth                             | /ExtendedServicesService  |

| Service Name       | Description   | Service Endpoint          |
|--------------------|---|---------------------------|
|                    | Science services, interfaces, GUIs, and advertisements  |                           |
| Group Management   | Data Partner service to organize users into groups for data access control and notification   | /GroupManagementService   |
| Invocation         | Simple service brokering capabilities that allow ECHO to execute external service requests asynchronously on behalf of users                | /InvocationService        |
| Invocation Utility | Wrapper service to allow external Earth Science services to post status, execute asynchronously and return results to users                 | /InvocationUtilityService |
| Order Management   | Service to create and submit orders for users   | /OrderManagementService   |
| Order Processing   | Data Partner -oriented service to fill and apply a status to user orders  | /OrderProcessingService   |
| Provider           | Data Partner account creation and maintenance   | /ProviderService          |
| Status             | General status information of asynchronous user requests—currently only supported by the Invocation Service and Invocation Utility Service. | /StatusService            |
| Subscription       | User service for creating data subscriptions so as to be notified when data is updated or modified  | /Subscription             |

| Service Name | Description   | Service Endpoint |
|--------------|---|------------------|
| Taxonomy     | Management interface for data and service classification schemes used by ECHO, Data Partner and Client Partners | /TaxonomyService |
| User         | User account creation and maintenance   | /UserService     |

## ECHO Error Handling

The ECHO 9.0 Web Service API has advanced error-reporting capabilities. Refer to Appendix E for a list of specific faults and their meanings.

## ECHO Globally Unique Identifiers (GUIDS)

An ECHO Globally Unique Identifier (GUID) is a mostly random number with a large number of unique keys. A GUID is normally a 16-byte (128-bit) number in hexadecimal form.

ECHO uses GUIDs to identify items such as users, providers, contacts, orders, etc. Client applications use GUIDs to find and operate on items using the ECHO API. Client applications should avoid displaying a GUID to the end user, as it is a machine-to-machine identifier. Most of the items in ECHO also accommodate a name preferred by the user; the client application can simply map from the item's user-displayed name to its actual GUID.

In almost all cases (with the notable exception of the Taxonomy API), the GUID on an item should be null when the item is sent to ECHO to be created. Once ECHO creates the item, it will generate a new GUID for the item and return it to the client. You can retrieve a list of **NameGuids** for most items. **NameGuids** provide the client a name-to-GUID mapping of all of the items in question. This can be very useful for displaying a list of items to the user and allowing the user to select an item of interest.

## ECHO Entities

This section describes several high-level concepts that help you understand the ECHO system. The following is a selected list of entities. For the complete list of ECHO entities, refer to <http://www.echo.nasa.gov/reference/reference.shtml>.

### Users

The most basic entity in the ECHO system is a **user**. Each user is identified by a unique user name. There are two types of users: **registered users** and **guests**. Registered users can save information they plan to use in their next session. Guests have the ability to do many of the things registered users can do, but they cannot count on persistent access to information across sessions in addition to other limitations.

## Roles

ECHO regulates access privileges based on the concept of user roles. User roles are a way to grant a user access to the system. These roles facilitate greater flexibility with operation-level authorization and allow certain users the ability to have more than one role without having more than one account in the system.

Some roles are always associated with a user account while others can be set dynamically by a client application. User roles are a way to grant a user access to the system. These roles facilitate greater flexibility with operation-level authorization and allow certain users the ability to execute both client and provider operations without having two accounts in the system. Currently, there are two kinds of roles:

1. **Provider Role:** A user can have one or more provider roles, with each role associated with one provider in the system. A user with one or more provider roles can access and update information about the providers with which they are associated. For instance, if a user has a provider role for Oak Ridge National Laboratory (ORNL), then that user can use the UpdateContact operation in ProviderAccountService to update the contact information for ORNL.

If the user is associated with one or more provider roles, you must tell the system which provider they currently want to represent. Thus, each user has a “provider context.” The provider context holds the current provider that user represents, and as the name implies, it is in the context of this provider to which the provider-oriented operations in the system are applied.

*Note: If the user is associated with only one provider role, the system assumes the provider associated with that one provider role is specified in the user’s “provider context.”*

2. **Admin Role:** A user can only be associated with one instance of this role. This role allows the user to access and update information about any user and any provider. Essentially, this role should allow the user full access to almost anything in the system and available through the current API. Users assigned this role should read the ECHO Operations Guide as well as this document.

## Queries

A **query** in the ECHO system is the term used for search and retrieval of science metadata stored by ECHO. You can search ECHO by calling the **ExecuteQuery** operation on the **CatalogService**—refer to “Chapter 4: Querying for Earth Science Metadata,” Page 23, for a detailed discussion of queries.

**NOTE:** You can search on collections (referred to as a “**Discovery Search**” or on granules (referred to as an “**Inventory Search**”).

ECHO supports storage and retrieval of previously stored queries by users. You can also execute a previously stored query to get data that are more recent.

ECHO can execute a query synchronously (not returning until the query completes) or asynchronously (returning immediately from the call and storing the results for later retrieval).

## Results

ECHO automatically generates a **ResultSet** after an ECHO query. Every result set in ECHO has a unique identifier (that is, a GUID) within the system.

You can execute multiple queries simultaneously. Result sets from all these queries will be available by name (**ResultSetGUID**); however, result sets may be removed from ECHO without notice in a reasonable amount of time, unless they were explicitly saved using the **SaveResultSets** operation. Since guests are not allowed to save results in ECHO, the results are not available after the guest completes the ECHO session. Results can be retrieved by using the **GetQueryResults** operation.

## Catalog Items

A **catalog item** is any metadata item (granule or collection) that is available for ordering from the ECHO system. The results of a query may return several granules or collections. Catalog items are identified by a **catalog item GUID** (CatalogItemId, which is an assigned XML metadata tag).

To see the possible required/optional options for a catalog item, invoke the **GetCatalogOrderInformation** operation on the **OrderManagementService** passing the catalog item GUIDs of interest.

If a URL is provided, you (Client Partners) may retrieve the data from that URL as a direct link or obtain the data accessing instruction via the URL.

## Orders

An **order** is a collection of **catalog** items that you, as a Client Partner, want to have and plan to order from a Data Provider. Each item in the order is associated with a quantity and any options available to that item. Within ECHO, a user creates an order and then adds, deletes, and updates each item in the order before submitting the order to ECHO. Orders can also be created and submitted in a single web service API call (CreateAndSubmitOrder). Registered Client Partners can look at the status of their orders, as well as the history of all their submitted and shipped orders.

The **collection** of catalog items that comprises an order does not have to belong to one provider, but can span many providers. When organizing providers and catalog items within an order, another concept called a “provider order” is used. An order can consist of one or more provider orders. Each provider order can consist of one or more catalog items that belong to the same provider. To identify a specific provider order, you need the GUID of the order that the provider order is in and the GUID of the provider that is associated with that provider order.

When a full order is submitted, ECHO splits the user’s order into separate provider orders and submits each provider order to the associated Data Provider.

The **OrderManagementService** allows users to create and change orders, provider orders, or individual catalog items. Once the **SubmitOrder** operation is executed for a certain order within the **OrderManagementService**, the user can no longer execute any further changes on that order. However, a registered user can look at the current and historical status of any of their submitted orders.

Once a provider order is submitted to the appropriate provider, the status of that order can be changed in two ways:

- The Data Provider can send an immediate response, whether they will or will not accept the order, to an order submission.
  - The Data Provider can wait and asynchronously use the **OrderProcessingService** to change the status of an order after they have had time to process the order.
- Order Options (Used by Data Partners)

Data Partners use **order options** to describe the structure of the data to request of the client as well as how to display the order form to the client and use ECHO Forms to perform both these functions. For details about ECHO Forms, refer to the ECHO Forms schema and other information at

[http://www.echo.nasa.gov/data\\_partners/data\\_tools.shtml](http://www.echo.nasa.gov/data_partners/data_tools.shtml).

**Option definitions** contain a name, scope, and deprecated flag, and an ECHO Form. Every option definition for a provider has a unique name. The **scope** indicates whether it is a system-level option definition or provider-level option definition. The **deprecated flag** indicates that you, as a Data Partner, have made that option definition obsolete. The deprecated flag indicates to clients that they should no longer use the option definition. The form part of the option definition contains an ECHO Form.

The **OptionSelection** is the data from the client for a specific option definition. The selection should be put in the content part of the option selection. Options are also used in ECHO to describe authenticators and options for ordering data from a provider.

### **Authenticators**

While most items that can be ordered through ECHO's data broker functionality are available to the entire user community, some items are restricted. Different Data Providers use different mechanisms to determine if the person ordering restricted data is allowed to access that data. Some providers use a password, others an authentication key (which is a string). To accommodate these needs, ECHO provides the authenticator mechanism. Providers are allowed to define authenticator structure using options. Clients can present the authenticator in a template form to the user. The client can then create a named list of authenticators on behalf of the user for each provider using operations in the user account service. ECHO allows for a registered user to have more than one authenticator stored for a provider since it is possible that different projects that the user represents may have different authenticators. There are operations for managing this list. Once the list is created, any client can set which one (by name of the authenticator) should be used for a given order. If none is set, then an authenticator is not used. ECHO will send the authenticator along with the order.

## Groups

The term **groups** refers to an aggregating mechanism in ECHO that allows Client Partners to associate a group name with a given set of users. When a group is created, the group's owner specifies an ECHO user to be the group's manager. However, group managers can be added and removed after creation by other group managers. After becoming a member of a group, a user can be granted access to restricted metadata via the Data Management Service.

## Extended Services

ECHO Extended Services allows partners to advertise and register web service interfaces, implementations, GUIs, and advertisements.

For more information about Extended Services and Service Partners, refer to the forthcoming *ECHO Service Partner's Guide*.

**This page is intentionally left blank.**



## Chapter 3: Logging In, Setting Up, and Getting Started

This section contains information and examples that are common to most ECHO client applications, such as basic Login and Logout and creation and management of user accounts.

### Creating and Managing ECHO Sessions

When using the Web Services API, you need to request an **ECHO token**. This token acts as your session key and must be passed to all other ECHO operations. All clients interfacing with the system are required to pass client identifier information to ECHO. The client identifier is a short description and/or name of the client. Client developers are encouraged to keep this information as concise as possible and to work with the ECHO Operations Group to create an appropriate identifier.

*NOTE: If client identifier information is not provided, submitted orders will fail.*

### Logging In to ECHO

Code Listing 1: Logging In as a Guest

```
// Client information is optional information provided by the
// client to ECHO when a user logs in
ClientInformation clientInfo = new ClientInformation();
clientInfo.setClientId("A Client");
clientInfo.setUserIpAddress("192.168.1.1");

// Call login with guest as username, email as password, and
// client information
String token =
    authenticationService.login("guest", "john@doe.com",
    clientInfo, null, null);
```

To obtain a token, call the Login operation of the Authentication Service. Below is an example of logging in to ECHO and creating a session token for a guest user.

#### Code Listing 2: Logging In as a Registered User

---

```
// Client information is optional information provided by the
// client to ECHO when a user logs in
ClientInformation clientInfo = new ClientInformation();
clientInfo.setClientId("A Client");
clientInfo.setUserIpAddress("192.168.1.1");

// Call login with jdoe as username, mypass as password, and
// client information
String token =
    authenticationService.login("jdoe", "mypass", clientInfo,
    null, null);
```

---

#### Logging Out of ECHO

When you are finished with a token, for example, you have finished a session with ECHO, destroy the token using the Logout operation.

#### Code Listing 3: Logging out of ECHO

---

```
// Logout and set token to null because it is not useful anymore
authenticationService.logout(token);
token = null;
```

---

You do not need to destroy a token after each call; you may reuse a token until it is destroyed with the Logout operation or the token expires.

*Because the token is used to track your session, it must be protected by client applications with the same level of security that you use for your login name and password.*

## Interacting with ECHO

Interacting with the rest of the ECHO API follows the same pattern as logging in and logging out except that it requires that you pass a valid ECHO token to each operation. The following example shows logging in, retrieving the version number of the ECHO system and logging back out.

Code Listing 4: Getting the ECHO Version Number

---

```
// Login
String token =
    authenticationService.login("jdoe", "mypass",
        new ClientInformation("A Client", "192.168.1.1"), null,
        null);

// Print out echo version number.
System.out.println("ECHO's version number: "
    + authenticationService.getECHOVersion(token));

// Logout using token from previous login
authenticationService.logout(token);
token = null;
```

---

## User Accounts

Use the **UserService** to update information such as addresses, e-mail addresses and phone numbers associated with a particular user. A feature of the ECHO system allows you the option to store multiple addresses and phone numbers distinguished by unique names, such that you can later choose from among them. For example, you may set up an address in your profile where the AddressName value is “Work” and set up another address where the AddressName value is “Home.” The API does not explicitly connect these, but a client interface can query for a list of addresses and fill in the blanks in the other API operations appropriately.

### Addresses

You are only required to submit address information to the ECHO system if you plan to order data, and then you are required to submit a shipping address, a billing address or a contact address.

An ECHO Address entity consists of an address name, a U.S. format flag, five street address lines, and city, state, zip code and country fields. The address name and address lines are required fields.

The address name represents a unique name for the address entity like “Home” or “Work”.

The U.S. format flag defaults to the setting “TRUE” which displays standard U.S. city, state, zip code and country address components as separate, required fields. Since this predictable address format is not the standard outside the U.S., you may set the U.S. format flag to “FALSE” so as to display five free-form address lines for international

users and providers to complete as appropriate. In either instance, the country field is required and must comply with the ISO three-letter country code. For example, “USA” stands for the United States of America while “CAN” stands for Canada.

### **E-mail**

Only one e-mail address can be associated with a user account. The rules for an e-mail address in the ECHO system are consistent with global standards for e-mail addresses: username@domain.

### **Phone Numbers**

As with the Address entity, you may store multiple phone numbers in the Phone Number entity. Each phone number is assigned with a **PhoneNumberType** from a list of common phone number types like HOME or BUSINESS. **CustomTypeName** can be used if one of the phone number types available does not meet your needs. The actual phone number is stored as a string in **Number**.

### **User Domain and User Region**

User Domain indicates the domain you are in such as government, commercial, or education. User Region specifies whether you are located in the U.S. or internationally.

### **Password**

ECHO requires your password to have at least 10 characters and at least three of the following types of characters:

- Upper case letters
- Lower case letters
- Digits
- Any other special characters such as “\$”, “&”, “>”, “<”, etc.

### **Creating a User Account**

To create a new user account with ECHO, fill in the User object including the phone numbers and other user attributes described above, and pass the object to the **CreateUser** operation on the **UserService**.

*All Globally Unique Identifiers (GUIDs) should be set to null when creating a new user. The system will create and assign GUIDs when it creates the account and return the user GUID from the operation.*

You may be logged in as a guest or authenticated user when creating a new user account.

#### Code Listing 5: Creating an ECHO Account

---

```
// Create work address for user
Address address = new Address();
address.setAddressName("Work");
address.setStreet1("123 Main Street");
address.setCity("Baltimore");
address.setCountry("USA");
address.setState("MD");
address.setUsFormat(true);
address.setZip("12345");

// Create business phone
Phone phone = new Phone();
phone.setNumber("443-555-5555");
phone.setPhoneNumberType(PhoneType.BUSINESS);

// Create user object
User user = new User();
user.setDomain(UserDomain.K12);
user.setRegion(UserRegion.USA);
user.setFirstName("John");
user.setLastName("Doe");
user.setOrganizationName("EarthSpace Inc.");
user.setEmail("jdoe@earthspace.com");
user.setPhones(new Phone[] { phone });
user.setAddresses(new Address[] { address });

// Use a strong password for user
String password = "K93FeS3e";

// Create user in ECHO
String userGuid = userService.createUser(guestToken, password,
user);
```

---

Once you have a valid user name and password, you can login to ECHO.

**This page is intentionally left blank.**

## Chapter 4: Querying for Earth Science Metadata

ECHO uses the **ECHO Alternative Query Language (AQL)** for its querying capabilities. (See Chapter 5 for a detailed explanation of the language syntax.) To support potentially large queries, ECHO handles a query expression without regard to the details of whether and how it will return query results to the user.

An ECHO query consists of a query expression specified in XML, starting with <query>. All AQL queries must conform to the AQLQueryLanguage.dtd available on the ECHO website. Also, refer to “Chapter 5: The ECHO Alternative Query Language” on Page 47 for additional information about AQL.

*You should validate your query against the DTD **before** passing it to ECHO. Since a query is passed to ECHO as a string, the query will not be validated against the DTD by the Web Service processing layer but rather at execution time, which, in the case of an asynchronous query or a subscription, could be after the Web Service call has completed.*

You may use the optional argument **MaxResults** to limit the numbers of results returned from a query, which is useful when a query produces more data than can be processed. This also saves processing time for the query and presentation of results. ***ECHO does not currently support the MaxResults function in a spatial query.***

You may retrieve the results of a query in several different ways. The **ExecuteQuery** operation on the Catalog Service allows you to indicate whether you would like the data itself returned or simply the number of hits. You can specify this function in the **ResultType** argument with one of the following values:

| Value   | Description  |
|---------|--|
| RESULTS | Specifies that you want the results of the query returned as part of the response. When using this option, you may choose to limit the actual metadata values returned. In addition, ECHO will return a result set identifier (ResultSetGUID) for subsequent retrievals of the results or for paging through the result list. <b><i>Note that ECHO Operations limits the maximum number of items returned to 2,000 at a time.</i></b> The complete results are stored in your result set which you can retrieve by using the <b>GetQueryResults</b> operation. |

| Value             | Description   |
|-------------------|---|
| RESULT_SET_GUID   | Specifies that you want the response to return only a key (ResultSetGUID). ECHO will generate a result set but will not return any results. You must subsequently retrieve the results using the <b>GetQueryResults</b> operation.  |
| HITS              | Specifies that you want the response to return both the ResultSetGUID and the number of records in the result set. ECHO will generate a result set but will not return any results. The number of records may be a <b>statistically</b> determined for large result sets. You must subsequently retrieve the results using the <b>GetQueryResults</b> operation.  |
| ITEM_GUIDS        | <p>Specifies that both the total number of item GUIDs searched and the total number of item GUIDs that matched the query criteria should be returned directly. <i><b>Note: No ResultSetGUID is returned since results do not persist in the system.</b></i></p> <p>All the GUIDs of the granules/collections that satisfy the query are returned to the client. It is the client's responsibility to request the metadata for each individual granule/collection using the <b>GetCatalogItemMetadata</b> operation discussed later.</p>   |
| NUMBER_OF_RESULTS | Specifies that you only want the response to return the number of records in a result set. ECHO will not, however, create or store the actual result set. The number of records may be statistically determined for large result sets. Use this result type when your only interest is in the number of records that match the query.   |
| ASYNCHRONOUS      | <p>Specifies that you want the query to run asynchronously. The benefit of an asynchronous query is that ECHO will return a response message immediately, without waiting for the completion of the query. Use this result type for queries that may require a long time to complete. ECHO will include the ResultSetGUID in the response message, which serves as a reference to the query.</p> <p>You may access intermediate results of an asynchronous query with the <b>GetQueryResults</b> operation, using the ResultSetGUID. You may cancel an asynchronous query using <b>CancelAsynchronousQuery</b> operation.</p> |



## Formatting Your Query Results

You can specify a subset of the information in the result set by using different parameters for the operations **ExecuteQuery**, **ExecuteSavedQuery**, and **GetQueryResults**.

The following elements are used to specify the format and content of a result set:

| Argument           | Description  |
|--------------------|--|
| IteratorSize       | Specifies the number of results to be returned from a single operation. This does not limit the number of items a query may match (see MaxResults) but limits to 2,000 the number of matching items returned in the result set, starting from the Cursor position. |
| Cursor             | Specifies the index of the first record to be returned in the result set. (For example, a value of 5 will return results starting from the fifth record. If none is specified, it defaults to 1). If you repeat the same query later, use the same Cursor value.   |
| MetadataAttributes | Specifies which fields of the ECHO Metadata you actually want to return. By only requesting the parts of the metadata you are interested in, you can increase query performance substantially. By default ECHO returns all of the metadata for each item.          |

Metadata results are returned as XML that conforms to one of the following DTDs:

- Granule metadata conforms to the Granule Results DTD—refer to “Appendix E: Results DTDs,” Page 109 (also located at: <http://www.echo.nasa.gov/reference/reference.shtml>).
- Collection metadata conforms to the Collection Results DTD “Appendix E: Results DTDs,” Page 109 (also located at: <http://www.echo.nasa.gov/reference/reference.shtml>).

Metadata attributes are made up of two values: the XML metadata attribute name and a primitive type name. ***ECHO currently ignores the type name.*** The allowable metadata attribute names are specified in the appropriate DTD (ECHOGranuleResults.dtd for granules and ECHOCollectionResults.dtd for collections). If you specify a metadata attribute name that has sub-attributes, all of the sub-attributes will be included as well. For example, if you specify **Platform**, the following elements will be included in the metadata:

- Platform
- PlatformShortName
- Instrument
- InstrumentShortName
- Sensor
- SensorShortName
- SensorCharacteristics
- SensorCharacteristicName
- SensorCharacteristicValue
- OperationMode

When you specify a sub-attribute, ECHO will return the “parent” attribute in the hierarchy as well as the sub-attribute. This allows you to ensure that data are correctly scoped. For example, if you specify **Sensor**, the following elements will be included in the metadata:

- Platform
- PlatformShortName
- Instrument
- InstrumentShortName
- GranuleUR
- GranuleURMetaData

***Detailed spatial attributes cannot be used as MetadataAttributes; only their containing element may be specified.*** For example, you cannot use **BoundingBox** as a MetadataAttribute, but you can use **HorizontalSpatialDomainContainer**. The following spatial elements cannot be specified as MetadataAttributes:

- Point
- Circle
- BoundingRectangle
- GPolygon
- Polygon
- PointLongitude
- PointLatitude
- CenterLongitude
- CenterLatitude
- Radius

- WestBoundingCoordinate
- NorthBoundingCoordinate
- EastBoundingCoordinate
- SouthBoundingCoordinate
- Boundary
- ExclusiveZone
- SinglePolygon
- MultiPolygon
- OutRing
- InnerRing

Specifying GranuleURMetaData as a MetadataAttribute is equivalent to not specifying any MetadataAttributes; the result set includes all the elements in the result DTD.

The following code snippet shows how to execute a query for all of the metadata for matching items.

**Code Listing 6: Executing a Simple Query**

---

```
// Execute a query to get results
QueryResponse response =
    catalogService.executeQuery(userToken, queryString,
        ResultType.RESULTS,
        10, // Iterator
        0, // Cursor
        3000, // max results
        null); // no metadata attributes specified
```

---

The following code snippet shows the same query but restricts the returned information to just the spatial container information.

**Code Listing 7: Executing a Query for Specific Metadata**

---

```
MetadataAttribute[] attributes =
    new MetadataAttribute[] { new MetadataAttribute(
        "HorizontalSpatialDomainContainer", null) };

QueryResponse response =
    catalogService.executeQuery(userToken, queryString,
        ResultType.RESULTS,
        10, // Iterator
        0, // Cursor
        3000, // max results
        attributes);
```

---

## Saving Complex Queries

To support creation and reuse of complex queries across clients, ECHO supports the ability to save a query for later execution. ECHO saves a query in its AQL form, to be re-evaluated each time the query is executed. You can save a persistent query by simply calling the **SaveQuery** operation and passing in an AQL query in the same format as you would pass it to **ExecuteQuery**, along with a name for your saved query. ECHO will return a GUID to use when trying to identify the query in the future.

### Code Listing 8: Saving a Complex Query

---

```
String queryGuid =
    catalogService.saveQuery(userToken, "Complex Spatial Query",
                             complexQueryString);
```

---

You can ask ECHO to execute a saved query by calling the **ExecuteSavedQuery** operation and passing your saved query GUID along with the same options as you would pass to **ExecuteQuery** as shown below.

### Code Listing 9: Executing a Saved Query

---

```
QueryResponse response =
    catalogService.executeSavedQuery(userToken, queryGuid,
                                     ResultType.RESULTS, 10, // Iterator
                                     0, // Cursor
                                     3000, // max results
                                     null); // no metadata attributes specified
```

---

You can see a list of the names of saved queries and their GUIDs by calling the **GetSavedQueryNames** operation as shown below.

### Code Listing 10: Getting the Names of Saved Queries

---

```
// Get saved query names
NameGuid[] savedQueryNameGuids =
    catalogService.getSavedQueryNames(userToken, queryGuids);

// print out query names
for (int i = 0; i < savedQueryNameGuids.length; i++)
{
    NameGuid guid = savedQueryNameGuids[i];
    System.out.println("Saved query name : " + guid.getName());
}
```

---

You can remove a saved query by calling the **RemoveSavedQueries** operation and passing the GUIDs of the queries to be removed, as shown.

**Code Listing 11: Removing Saved Queries**

```
// Remove the saved queries using the guids of the queries
catalogService.removeSavedQueries(userToken, queryGuids);
```

*At this point there are no restrictions on how many queries you may save. However, you should make an effort to keep the number of queries to a reasonable number.*

## Handling Large Result Sets

Given ECHO's large store of Earth Science data, it is possible for queries to return very large result sets. ECHO supports retrieving the results from a query in a number of ways. The simplest is to ask ECHO to return the results directly from the **ExecuteQuery** request by passing **RESULTS** as the **ResultType**. However, to prevent a single query from monopolizing ECHO resources, ECHO limits the number of results available in response to a query. By default, this limit is 2,000 items, though ECHO Operations may change it depending on ECHO usage patterns.

For larger results, ECHO supports a paging mechanism. This allows you to page through the available data in page sizes that you select (up to the ECHO Operations configurable limit). For **ResultTypes** of **RESULTS**, **RESULT\_SET\_ID**, **HITS**, and **ASYNCHRONOUS**, ECHO will create and store a result set and return the corresponding GUID. You can page through the result set using the **GetQueryResults** operation. The arguments to **GetQueryResults** are similar to **ExecuteQuery** with the exception of needing to specify the result set ID rather than a new query.

Result sets may change after they are created. Providers are continually changing the data they have registered in ECHO. New records are never added to a result set but records may be removed from a result set. Because of this, you should watch the fields **Cursor** and **CursorAtEnd** when paging through a large result set:

**Cursor** specifies the index of the first record to be returned in the result set. (For example, a value of 5 will return results starting from the fifth record. If none is specified, it defaults to 1). If you repeat the same query later, use the same **Cursor** value.

Use **CursorAtEnd** to determine when you have reached the end of the result set. This boolean field is true if the returned results were the last available results in the result set.

The following code illustrates paging through a result set and displaying it to the user.

**Code Listing 12: Paging Through Query Results.**

---

```
final int ITERATOR_SIZE = 10;
try
{
    CatalogServiceLocator catalogServiceLocator =
        new CatalogServiceLocator();
    CatalogServicePort catalogService =
        catalogServiceLocator.getCatalogServicePort();
    QueryResponse response =
        catalogService.executeQuery(userToken, userQuery,
            ResultType.RESULT_SET_GUID, 0, 0, 1000, null);
    String resultSetGuid =
        response.getResults().getResultSetGuid();

    // begin paging through results
    int cursor = 1;
    boolean atEnd = false;

    while (!atEnd)
    {
        //Get next ITERATOR_SIZE results
        QueryResults results =
            catalogService.getQueryResults(userToken,
                resultSetGuid, null, ITERATOR_SIZE, cursor);

        //Print out results
        System.out.println(results.getReturnData());

        //Set cursor to next index
        cursor = results.getCursor();

        //Check if at end of result set
        atEnd = results.isCursorAtEnd();
    }
    System.out.println("All results retrieved");
}
catch (EchoFault e)
{
    e.printStackTrace();
}
catch (ServiceException e)
{
    e.printStackTrace();
}
catch (RemoteException e)
{
    e.printStackTrace();
}
```

---

*Like **ExecuteQuery**, **GetQueryResults** takes an array of **MetadataAttributes**. Internally ECHO only stores in a result set the item IDs that match a given query. This means that you may pull different metadata from a single result set with each call by varying what you pass to the **MetadataAttribute** array without needing to re-query ECHO. It is highly recommended you use the **MetadataAttribute** array to restrict the information ECHO returns and thus improve performance.*

## Saving Results

ECHO retains query result sets for each user for a certain length of time according to ECHO Operations policy. You can indicate that you do not want a result set deleted as part of this routing maintenance by saving the result set. Saving a result set is simply a matter of calling the **SaveResultSets** operation and passing the GUIDs of the result sets to save, as shown below.

**Code Listing 13: Saving a Result Set for Future Use**

```
resultSetGuid = "[guid of result set from previous query]";
NameGuid[] resultSetNameGuids = new NameGuid[1];
resultSetNameGuids[0] = new NameGuid("My Example Results",
resultSetGuid);

// Ask ECHO to save the result set
catService.saveResultSets(token, resultSetNameGuids);
```

You can remove a saved result set by calling **RemoveSavedResultSets** and passing the GUIDs of the result sets to remove.

**Code Listing 14: Removing Saved Result Sets**

```
String[] resultSetGuids =
    new String[] { "[guid of result set from previous query]" };

// Ask ECHO to remove the result sets
catService.removeSavedResultSets(token, resultSetGuids);
```

## Visibility of Results

When you execute a query, the query is applied to all the data in ECHO to create the result set. However, when the results are retrieved, you may not see all of the items. What you can see depends on the rules defined by the Data Partners and the privileges granted to you.

## Restricted Items

If a particular item in your result set is restricted to you, based on your privileges, it will be removed from your result set.

## Deleted Items

When you execute a query, all matching items are saved within ECHO as the result set of the query. You can come back at a later time to view this result set (provided it has not been deleted from ECHO). It is possible that, between the time you execute a query and the time you view the results, some of the matched items might be deleted from ECHO or restricted due to a request from the Data Partner who owns the metadata. In that case, the item will be removed from the result set. For more information about notification of deleted or restricted order items, refer to “Restricted or Deleted Order Items,” Page 89.

## Querying for Orderable Data

In ECHO 9.0 you can exclude from your query data which cannot be ordered. Refer to “Orderable Items,” Page 65.

## Sample Queries

The following are sample queries that you can execute against ECHO. Note that the provider and the datasets used in these samples are representative only; you should modify the query to suit your needs.

### Code Listing 15: Sample Collection Query (Discovery Search)

---

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v9.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
  Search for collections from ORNL_DAAC that have
  parameter value that contains 'IMAGERY'
-->
<query>
  <for value="collections"/>
  <dataCenterId>
    <list><value>ORNL_DAAC</value></list>
  </dataCenterId>
  <where>
    <collectionCondition>
      <parameter>
        <textPattern>'%Imagery%'<</textPattern>
      </parameter>
    </collectionCondition>
  </where>
</query>
```

---



#### Code Listing 16: Sample Collection Query from Two Providers (Discovery Search)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v9.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/IIMSACLQueryLanguage.dtd">
<!--
  Search for collections from GSFCECS and ORNL_DAAC that have
  processing level 1A or 2
-->
<query>
  <for value="collections"/>
  <dataCenterId>
    <list>
      <value>GSFCECS</value>
      <value>ORNL_DAAC</value>
    </list>
  </dataCenterId>
  <where>
    <collectionCondition negated="y">
      <processingLevel>
        <list>
          <value>'1A'</value>
          <value>'2'</value>
        </list>
      </processingLevel>
    </collectionCondition>
  </where>
</query>
```

#### Code Listing 17: Sample Collection Query with Temporal and Spatial Constraints (Discovery)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v9.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/IIMSACLQueryLanguage.dtd">
<!--
Search for collections from ORNL_DAAC with:
temporal range: periodic range between Jan 1, 1990 and Dec. 31
1998from the 1st to the 300th day of each year, AND
spatial extent: bounding box 60S, 70W to 60N, 70E.
-->
<query>
  <for value="collections"/>
  <dataCenterId>
    <value>ORNL_DAAC</value>
  </dataCenterId>
  <where>
    <collectionCondition>
      <temporal>
        <startDate>
          <Date YYYY="1990" MM="01" DD="01"/>
        </startDate>
        <stopDate>
          <Date YYYY="1998" MM="12" DD="31"/>
        </stopDate>
        <startDay value="1"/>
      </temporal>
    </collectionCondition>
  </where>
</query>
```

---

```

        <endDay value="300"/>
    </temporal>
</collectionCondition>
<collectionCondition negated="n">
    <spatial operator="RELATE">
        <IIMSPolygon>
            <IIMSLRing>
                <IIMSPoint long='-10' lat='85' />
                <IIMSPoint long='10' lat='85' />
                <IIMSPoint long='10' lat='89' />
                <IIMSPoint long='-10' lat='89' />
                <IIMSPoint long='-10' lat='85' />
            </IIMSLRing>
        </IIMSPolygon>
    </spatial>
</collectionCondition>
</where>
</query>

```

---

**Code Listing 18: Sample Collection Query with Complex Temporal and Spatial Conditions (Discovery)**

---

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v9.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for collections from ORNL_DAAC with
temporal range: periodic range between Jan 1, 1990 and Dec. 31
1998
from the 1st to the 300th day of each year, AND
some days of January.
source name: L7 or AM-1 AND
spatially covering any 'temperate' region or USA
-->
<query>
    <for value="collections"/>
    <dataCenterId>
        <list><value>ORNL/value</list>
    </dataCenterId>
    <where>
        <collectionCondition>
            <temporal>
                <startDate>
                    <Date YYYY="1990" MM="01" DD="01"/>
                </startDate>
                <stopDate>
                    <Date YYYY="1998" MM="12" DD="31"/>
                </stopDate>
                <startDay value="1"/>
                <endDay value="300"/>
            </temporal>
        </collectionCondition>
        <collectionCondition negated='n'>
            <sourceName>
                <list>

```

---

---

```

        <value>'L7'</value>
        <value>'AM-1'</value>
    </list>
</sourceName>
</collectionCondition>
<collectionCondition>
    <spatialKeywords>
        <list>
            <value>'temperate'</value>
            <value>'USA'</value>
        </list>
    </spatialKeywords>
</collectionCondition>
<collectionCondition>
    <temporalKeywords>
        <textPattern>'%january%'</textPattern>
    </temporalKeywords>
</collectionCondition>
</where>
</query>

```

---

**Code Listing 19: Sample Collection Query Using Provider Specific Attributes (Discovery)**

---

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v9.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/IIMSACLQueryLanguage.dtd">
<query>
<for value="collections"/>
<dataCenterId>
    <value>ORNL_DAAC</value>
</dataCenterId>
<where>
    <collectionCondition>
        <additionalAttributeNames>
            <list>
                <value>'Provider_Specific_Attribute_1'</value>
                <value>'Provider_Specific_Attribute_3'</value>
            </list>
        </ additionalAttributeNames >
    </collectionCondition>
</where>
</query>

```

---

## Using Subscriptions to Automate Queries

The Subscription Service allows you to subscribe to receive automatic notification when providers add, update, or delete metadata. This capability can be extremely valuable for client developers, because it provides the opportunity to set up post-processing and data massaging that runs when new or changed metadata is delivered, in order to convert it into an application-specific form.

The three key questions in creating a subscription are:

- ***What dataset are you interested in?*** To determine what providers exist in the system, you can call the **GetProviderNames** operation on the Provider Service, passing null for the GUIDs. Additionally, the Catalog Service discovery mechanism retrieves the datasets currently stored in the metadata repository.
- ***What type of metadata from the dataset are you interested in?*** ECHO allows you to subscribe to collection-level metadata, granule-level metadata, or both. Collection metadata will rarely change whereas granule metadata will be frequently updated or created; most users choose to subscribe to granule metadata for this reason. Be aware that providers may also delete granule metadata from time to time.
- ***Where should ECHO send the metadata when it is updated?*** ECHO supports two methods of subscription delivery: FTP and e-mail. Due to security restrictions, ECHO requires all FTP deliveries to be transferred via passive mode. If delivery is not possible (because the FTP server is full or the mail server rejects the large file attachment), ECHO sends an e-mail notification to the subscriber. Consult your local administrator for more information on passive-mode transfers and file size limitations.

In addition to the basic subscription information listed above, you may provide an AQL query to the Subscription Service that will be executed on your behalf when your subscription is triggered. In this case, the results of your query are sent rather than the raw updated metadata.

### Creating a Subscription

To create a subscription, call **CreateSubscriptions** and pass in the detailed **MetadataSubscription** information as described in the API documentation. Note that the subscription GUID should be null since you are requesting a new subscription be created. **CreateSubscriptions** will return a list of GUIDs to identify the newly created subscriptions. The GUIDs will be listed in the same order in which you supplied the **MetadataSubscription** information.

#### Code Listing 20: Creating a Basic Subscription

---

```
String token = "[token obtained from login]";

// Get the subscription service
SubscriptionServiceLocator locator = new
    SubscriptionServiceLocator();
SubscriptionServicePort subService =
    locator.getSubscriptionServicePort();

// The information we want to subscribe to
String providerGuid = "[provider guid obtained using
    ListAllProviders]";
String datasetName = "dataset name obtained through a query";
String query = "[AQL query assembled to limit the data
    returned]";

// Filtering information
MetadataFilterInfo filterInfo = new MetadataFilterInfo(query, 5);
MetadataActionInfo actionInfo =
    new MetadataActionInfo(null, CompressionType.GZIP,
        SubscriptionUpdateType.COLLECTIONS_ONLY);

// Delivery information
DeliveryInfo deliveryInfo =
    new DeliveryInfo(DeliveryType.EMAIL,
        "example@example.org", null, null,
        20);
Calendar stopTime =
    Calendar.getInstance(TimeZone.getTimeZone("GMT"));
stopTime.add(Calendar.YEAR, 1);

// Package it up in a MetadataSubscription element
MetadataSubscription[] metadataSubscriptions = new
    MetadataSubscription[1];
metadataSubscriptions[0] =
    new MetadataSubscription(null, null, "Example Simple
    Subscription",
        providerGuid, datasetName, filterInfo, actionInfo,
    deliveryInfo,
        stopTime);

// Ask ECHO to create the subscription
String[] guids =
    subService.createSubscriptions(token, metadataSubscriptions);
```

---

*If you provide a temporal condition in an AQL query for your subscription, you must leave **NumberOfDays** empty in **MetadataFilterInfo**. Likewise, if you use **NumberOfDays**, you may not have a temporal constraint in your AQL.*

Each subscription must have the following information:

Table 2: Subscription Information

| Subscription Parameter   | Description   |
|--------------------------|---|
| Subscription Name        | A unique name to describe this subscription. In ECHO 9.0 the name must be unique.   |
| <i>Provider GUID</i>     | The unique ID of the provider you wish to subscribe to. If you want to subscribe to all providers, you may use an ‘*’ with constraints.   |
| Dataset Name             | The name of the dataset you wish to subscribe to or ‘*’ to indicate all datasets from the selected provider(s).   |
| Metadata Filter Info     | <p>The spatial and/or temporal constraint to use for the subscription to narrow the data to a specific geographic location or period(s) of time.</p> <p>The AQL query filter is used to specify an IIMS AQL condition when validating metadata updates specified in your subscription preferences.</p> <p>The temporal filter element is an integer that represents a specified number of days prior to today’s date, that is, a span of days, which allows you to focus on data whose acquisition dates fall within that span.</p> <p>Note that you can only set the temporal filter in either the AQL <b>TemporalCondition</b> or in <b>NumberOfDays</b>. It is an ERROR to set it in both.</p> |
| Metadata Action Info     | The information related to processing of the subscription. Like Query, you can limit the metadata fields returned by a subscription by specifying a list of <b>MetadataAttributes</b> .   |
| Compression Type         | The compression type to use on the metadata file. Note that the subscription files delivered from ECHO are XML-formatted and highly compressible. It is recommended that you use a <b>CompressionType</b> of GZIP to conserve space and bandwidth.  |
| Subscription Update Type | The type of metadata to subscribe to:   |

| Subscription Parameter | Description  |
|------------------------|--|
|                        | <p>1. <b>ALL_COLLECTIONS</b>: You will receive all updated collection metadata.</p> <p>2. <b>COLLECTIONS_ONLY</b>: You will receive all updated collection metadata from within your specified collection.</p> <p>3. <b>GRANULES_ONLY</b>: You will receive all updated granule metadata from within your specified collection.</p> <p>4. <b>BOTH</b>: You will receive both the updated collection metadata and the updated granule metadata from within your specified collection.</p>   |
| Delivery Info          | <p>The way in which metadata should be delivered from the subscription: <b>EMAIL</b> or <b>FTPPUSH</b>.</p> <p>If e-mail is chosen, the <b>DeliveryAddress</b> should be of the format “username@domain”.</p> <p>If FTP Push is chosen, the <b>DeliveryAddress</b> should also be of the format “username@ftp.domain”. For FTP deliveries, the <b>DeliveryFolder</b> and <b>Password</b> fields are required.</p> <p>The Subscription Service has a governor that prevents delivery of metadata above a certain threshold. <b>LimitSize</b> is your specified maximum acceptable size (in megabytes) of the delivery file. The system calculates the size of the data file prior to delivery; if the size exceeds <b>LimitSize</b>, the file is not delivered.</p> |
| StopTime               | The time after which metadata should cease to be delivered.  |

## Deleting a Subscription

To remove a subscription, call the **RemoveSubscriptions** operations with the GUIDs of the subscriptions to remove, as shown below.

### Code Listing 21: Removing a Subscription

---

```
String[] subGuids = new String[] { "[subscription guid]" };
subService.removeSubscriptions(token, subGuids);
```

---

## Listing Subscriptions

You can list the names and GUIDs of subscriptions using the **GetSubscriptionNames** and **GetSubscriptionNamesByState** operations.

### Code Listing 22: Listing Subscriptions

---

```
subGuids = new String[] { "[subscription guid]" };
NameGuid[] allActiveNames =
    subService.getSubscriptionNames(token, null);

NameGuid[] specificActiveNames =
    subService.getSubscriptionNames(token, subGuids);

NameGuid[] allExpiredNames =
    subService.getSubscriptionNamesByState(token,
        SubscriptionState.EXPIRED);
```

---



### Code Listing 23: Sample Granule Query Using Various Granule Conditions (Inventory)

---

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v9.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for granules from the L70R or L70RWRS or
GLCF_GRANULE_METADATA datasets that have Browse data and were
categorized as day granules.
-->
<query>
<for value="granules"/>
<dataCenterId>
  <list>
    <value>ORNL_DAAC</value>
  </list>
</dataCenterId>
<where>
  <granuleCondition>
    <browseOnly/>
  </granuleCondition>
  <granuleCondition>
    <cloudCover>
      <range lower="10" upper="20"/>
    </cloudCover>
  </granuleCondition>
  <granuleCondition>
    <dataSetId>
      <list>
        <value>'L70R'</value>
        <value>'L70RWRS'</value>
        <value>'GLCF_GRANULE_METADATA'</value>
      </list>
    </dataSetId>
  </granuleCondition>
  <granuleCondition>
    <dayNightFlag value="DAY"/>
  </granuleCondition>
</where>
</query>
```

---

#### Code Listing 24: Sample Granule Query Using Spatial and Temporal Bounds (Inventory)

---

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v9.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/IIMSACLQueryLanguage.dtd">
<!--
Search for granules the specified spatial region, and periodic
temporal extent
-->
<query>
<for value="granules"/>
<dataCenterId>
  <list>
    <value>ORNL_DAAC</value>
  </list>
</dataCenterId>
<where>
  <granuleCondition>
    <spatial operator="RELATE">
      <Polygon>
        <LRing>
          <CList>-120, -30, -100, -60, 5, -90, 5, 85, -120, 30, -
120, -30</CList>
        </LRing>
        <LRing>
          <CList>80, 20, 80, 60, 20, 60, 20, 20, 80, 20</CList>
        </LRing>
      </Polygon>
    </spatial>
  </granuleCondition>
  <granuleCondition>
    <temporal>
      <startDate><Date YYYY="1989" MM="01" DD="01"/></startDate>
      <stopDate><Date YYYY="1998" MM="12" DD="31"/></stopDate>
      <startDay value="1"/>
      <endDay value="300"/>
    </temporal>
  </granuleCondition>
</where>
</query>
```

---

#### Code Listing 25: Sample Granule Query Using Sensor Name (Inventory)

---

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v9.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for granules from ORNL_DAAC with source name SWIR or TIR
-->
<query>
<for value="granules"/>
<dataCenterId>
  <list><value>ORNL_DAAC</value></list>
</dataCenterId>
<where>
  <granuleCondition>
    <sensorName>
      <list>
        <value>'SWIR'</value>
        <value>'TIR'</value>
      </list>
    </sensorName>
  </granuleCondition>
</where>
</query>
```

---

#### Code Listing 26: Sample Granule Query Using Temporal Constraints (Inventory)

---

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v9.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for granules with temporal range: periodic range between
Jan 1, 1990 and Dec. 31 1998 from the 1st to the 300th day
of each year.
-->
<query>
  <for value="granules"/>
  <dataCenterId>
    <list>
      <value>ORNL_DAAC</value>
    </list>
  </dataCenterId>
  <where>
    <granuleCondition>
      <temporal>
        <startDate>
          <Date YYYY="1990" MM="01" DD="01"/>
        </startDate>
        <stopDate><Date YYYY="1998" MM="12" DD="31"/></stopDate>
        <startDay value="1"/>
        <endDay value="300"/>
      </temporal>
    </granuleCondition>
  </where>
</query>
```

---

#### Code Listing 27: Sample Granule Query Using Additional Provider Specific Attributes (Inventory)

---

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v9.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
  <for value="granules"/>
  <dataCenterId>
    <value>ORNL_DAAC</value>
  </dataCenterId>
  <where>
    <granuleCondition>
      <additionalAttributes>
        <additionalAttribute>
          <additionalAttributeName>'COORDINATE_UNITS_NAME'</
additionalAttributeName>
          < additionalAttributeValue><value>'METERS'</value></
additionalAttributeValue>
          <additionalAttribute>
            </additionalAttributes>
          </granuleCondition>
        </where>
      </query>
```

---

**Code Listing 28: Sample Granule Query Using Additional Provider Specific Attributes with Complex Values (Inventory)**

---

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v9.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
<for value="granules"/>
  <dataCenterId>
    <value>ORNL_DAAC</value>
  </dataCenterId>
  <where>
    <granuleCondition>
      <additionalAttributes>
        <additionalAttribute>
          <additionalAttributeName>'SAMPLE_DATE'</
additionalAttributeName>
          <additionalAttributeValue>
            <dateRange>
              <startDate>
                <Date YYYY="2000" MM="05" DD="12"/>
              </startDate>
              <stopDate>
                <Date YYYY="2000" MM="10" DD="12"/>
              </stopDate>
            </dateRange>
          </additionalAttributeValue>
        </additionalAttribute>
      </additionalAttributes>
    </granuleCondition>
  </where>
</query>
```

---

## Chapter 5: The ECHO Alternative Query Language

ECHO Alternative Query Language (AQL) is a query language that defines the format for searches on collections (Discovery) and granules (Data or Inventory Search) in the ECHO system. AQL is an XML-based language. You can find the DTD for AQL at <http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd>

Code Listing 29: General AQL Structure

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v9.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
  <for value= "collections"/>
  // "granules" can be substituted for "collections"
  <dataCenterId>
    <all/>
  </dataCenterId>
  <where>
    <list of conditions ... >
  </where>
</query>
```

*Note: The <for/> tag shown in this example does not require that you set a value attribute; if you do not explicitly set this attribute, it will default to "collections." Your query will validate correctly, but you may still experience an error if you use granule settings for other attributes within the same query.*

You can specify the data centers that you wish to search by explicitly listing them or by using the empty element **<all/>** to indicate that you want all the data centers searched. This is the default condition, that is, if you do not specify a condition for the **dataCenterId** element, ECHO will search all data centers. *Do not enclose **dataCenterId** values in single-quotes.*

To search all data providers:

```
<dataCenterId><all/></dataCenterId>
```

To search for collections/granules in the ORNL data provider's metadata only:

```
<dataCenterId>
  <value>ORNL_DAAC</value>
</dataCenterId>
```

To search for collections/granules in the LPDAAC\_ECS data providers' metadata only:

---

```
<dataCenterId>
  <list>
    <value>ORNL_DAAC</value>
    <value>LPDAAC_ECS</value>
  </list>
</dataCenterId>
```

---

The list of conditions consists of **collectionCondition** elements for Discovery or **granuleCondition** elements for Inventory Search. The fields you can search on within collections occur as children of the **collectionCondition** element. The fields you can search on within granules occur as children of the **granuleCondition** element. ECHO will return only those granules or collections that satisfy all the conditions of the query, that is, ECHO applies the Boolean AND between all conditions of the query.

The **where** tag can have any number of **collectionCondition** or **granuleCondition** elements. Each of these tags can contain any one of the search elements listed in the Inventory Search and Discovery Search tables below. For the query to be valid, each of the search elements may appear only once.

Code Listing 30 shows the general structure of a Discovery query with multiple constraints.

**Code Listing 30: Structure of a Discovery Query with Multiple Collection Conditions**

---

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v9.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/IIMSACLQueryLanguage.dtd">
<query>
  <for value="collections"/>
  <dataCenterId>
    <value>ORNL_DAAC</value>
  </dataCenterId>
  <where>
    <collectionCondition>... </collectionCondition>
    <collectionCondition>... </collectionCondition>
    ...
  </where>
</query>
```

---



Code Listing 31 shows the general structure of an Inventory query with multiple constraints.

**Code Listing 31: Structure of an Inventory Query with Multiple Granule Conditions**

---

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v9.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/IIMSACLQueryLanguage.dtd">
<query>
  <for value="granules"/>
  <dataCenterId>
    <value>ORNL_DAAC</value>
  </dataCenterId>
  <where>
    <granuleCondition>... </granuleCondition>
    <granuleCondition>... </granuleCondition>
    ...
  </where>
</query>
```

---

If you specify a list or a single value, then the search looks for an exact match between the string specified and the data stored. For example, if the search value is 'Imagery' and the data stored has the value 'Satellite Imagery', the data will not be returned since there was no exact match. In this instance, it is more useful to use the **textPattern** element for the search.

## Discovery Search

Searches for collections must enclose the search criteria in a **collectionCondition** element. If you want to search the negated criteria, you should set the **negated** attribute in the **collectionCondition** element to 'Y'. Code Listing 32 shows how to create a search for all collections except the ones with processing level 1, 1A, 1B, or 2.

Code Listing 32: Using the Negated Attribute on Collection Conditions

```
<collectionCondition negated='Y'>
  <processingLevel>
    <list>
      <value>'1'</value>
      <value caseInsensitive='Y'>'1A'</value>
      <value>'1B'</value>
      <value>'2'</value>
    </list>
  </processingLevel>
</collectionCondition>
```

You can set the negated attribute to 'Y' for a criterion. Some search elements disallow use of the negated attribute; the definitions of the individual search elements beginning on Page 54 specifically state whether or not an element can use the negated attribute. *For those criteria that do not support the negated version, the criteria will remain positive even if you specify the attribute **negated='Y'** for the parent **collectionCondition** element.*

The following table lists the criteria you can use to search for collections. For details about each of these criteria, refer to the section “Common Search Elements” on Page 54.

Table 3: Collection Search Criteria

| Search Criteria         | XML Element                                |
|-------------------------|--|
| Campaign Short Name     | <CampaignShortName>...</CampaignShortName> |
| Dataset ID              | <dataSetId>...</dataSetId>                 |
| ECHO Insert Date        | <ECHOInsertDate>...</ECHOInsertDate>       |
| ECHO Last Update        | <ECHOLastUpdate>...</ECHOLastUpdate>       |
| Online Collections Only | <onlineOnly/>                              |
| Parameter               | <parameter>...</parameter>                 |
| Processing Level        | <processingLevel>...</processingLevel>     |
| Sensor Name             | <sensorName>...</sensorName>               |

| Search Criteria            | XML Element  |
|----------------------------|--|
| Short Name                 | <shortName>...</shortName>                               |
| Source Name                | <sourceName>...</sourceName>                             |
| Spatial                    | <spatial>...</spatial>                                   |
| Spatial Keywords           | <spatialKeywords>...</spatialKeywords>                   |
| Temporal                   | <temporal>...</temporal>                                 |
| Temporal Keywords          | <temporalKeywords>...</temporalKeywords>                 |
| Additional Attribute Names | <additionalAttributeNames>...</additionalAttributeNames> |
| Orderable Items            | <orderable/>   |
| Version ID                 | <versionId>...</versionId>                               |
| Archive Center             | <archiveCenter>...</archiveCenter>                       |
| Additional Attributes      | <additionalAttributes>... </additionalAttributes>        |

## Inventory Search

Searches for granules must enclose the search criteria in the **granuleCondition** element. As in the case of **collectionCondition**, if you want to search the negated criteria, set the **negated** attribute in the **granuleCondition** element to 'Y'. You can set the negated attribute to 'Y' for any criterion. Some search elements disallow use of the negated attribute; the definitions of the individual search elements beginning on Page 54 specifically state whether or not an element can use the negated attribute. *For those criteria that do not support the negated version, the criteria will remain positive even if you specify the attribute **negated='Y'** for the parent **granuleCondition** element.*

The following table lists the criteria you can use to search for granules.

**Table 4: Inventory Search Criteria**

| Search Criteria                                 | XML Element                                      |
|---|--|
| Only Granules with Browse Data                  | <browseOnly/>                                    |
| Campaign Short Name                             | <CampaignShortName>...</CampaignShortName>       |
| Percentage of cloud Cover                       | <cloudCover>...</cloudCover>                     |
| Dataset ID                                      | <dataSetId>...</dataSetId>                       |
| ECHO Insert Date                                | <ECHOInsertDate>...</ECHOInsertDate>             |
| ECHO Last Update                                | <ECHOLastUpdate>...</ECHOLastUpdate>             |
| Either Day or Night Granules Only               | <dayNightFlag/>                                  |
| Only Global Granules                            | <globalGranulesOnly/>                            |
| Search on ECHO granule IDs (formerly granuleId) | <ECHOGranuleID>...</ECHOGranuleID>               |
| Search on Granule UR (provider specific)        | <GranuleUR>...</GranuleUR>                       |
| Online Granules Only                            | <onlineOnly/>                                    |
| Two-D Coordinate System                         | <TwoDCoordinateSystem>...</TwoDCoordinateSystem> |
| Producer Granule ID                             | <ProducerGranuleID>...</ ProducerGranuleID>      |

| Search Criteria          | XML Element  |
|--------------------------|--|
| Additional Attributes    | <additionalAttributes>...</additionalAttributes>     |
| Sensor Name              | <sensorName>...</sensorName>                         |
| Source Name              | <sourceName>...</sourceName>                         |
| Spatial                  | <spatial>...</spatial>                               |
| Temporal                 | <temporal>...</temporal>                             |
| Orderable Items          | <orderable/>   |
| Version ID               | <versionId>...</versionId>                           |
| PGE Name                 | <PGENAME>...</PGENAME>                               |
| PGE Version              | <PGEVersion>...</PGEVersion>                         |
| Measured Parameters      | <measuredParameters>...</measuredParameters>         |
| Provider Production Date | <providerProductionDate>...</providerProductionDate> |
| Provider Insert Date     | <providerInsertDate>...</providerInsertDate>         |

*Effective with ECHO Release 7.0, the **LocalGranuleID** search criterion was renamed **ProducerGranuleID**.*

Refer to “Appendix E: Results DTDs,” Page 109 for complete DTD.

## Common Search Elements

You may use the following search criteria for both collection and granule searches.

### Case Insensitive Searching

Any **value** or **textPattern** tag has an attribute **caseInsensitive** that you can set to **Y** or **N** to indicate whether the search **value** or **textPattern** is case insensitive. The default is **N**, that is, the default is case sensitive searching.

### CampaignShortName

**CampaignShortName** is the name(s) of the campaign or project that gathered data associated with the collection or granule. In **CampaignShortName**, you can use a *single-quoted string* in a **value**, a list of *single-quoted strings* in a **list** or a **textPattern** element as defined in the section on Miscellaneous Elements on Page 78.

This example shows a search for collections/granules where **CampaignShortName** = 'River'.

---

```
<CampaignShortName>
  <value>'River'</value>
</CampaignShortName>
```

---

This example shows a search for collections/granules where **CampaignShortName** = 'River' or **CampaignShortName** = 'Lake'.

---

```
<CampaignShortName>
  <list>
    <value>'River'</value>
    <value>'Lake'</value>
  </list>
</CampaignShortName>
```

---

This example shows a search for collections/granules where **CampaignShortName** contains the string 'AS\_A%D', where the % represents zero or more numbers or letters between the *A* and the *D* in the text string 'AS\_A%D'

---

```
<CampaignShortName>
  <textPattern operator="LIKE">'AS_A%D'</textPattern>
</CampaignShortName>
```

---

### DataSetID

**dataSetID** specifies the universal name of a collection. You can use this element to restrict the search to a few collections, using the **value** element to specify a single collection, a **list** for a list of collections, or a **textPattern**. For the **value** and **list** elements, you must know the exact name of the collection. For example, you may have performed a Discovery search to locate the collection of interest and now want to perform an Inventory search for granules.

This example shows a search for all collections whose **dataSetId** ends with '1A'.

---

```
<dataSetId>
  <textPattern caseInsensitive="Y">%1A'</textPattern>
</dataSetId>
```

---

This example shows a search for all granules whose **dataSetId** is either 'LEAF CHEMISTRY, 1992-1993 (ACCP)' or 'ASTER DEM Product V002'.

---

```
<dataSetId>
  <list>
    <value>'LEAF CHEMISTRY, 1992-1993 (ACCP)'<</value>
    <value>'ASTER DEM Product V002'</value>
  </list>
</dataSetId>
```

---

### OnlineOnly

Use **OnlineOnly** to search for granules or collections that are available online. You cannot use the negated condition for this search.

---

```
<onlineOnly/>
```

---

### ECHO Insert Date

Use **ECHOInsertDate** to specify a search for collections or granules based on their insertion date into ECHO. You can only specify a dateRange. You cannot use the negated condition for this search.

---

```
<ECHOInsertDate>
  <dateRange>
    <startDate>
      <Date YYYY="2001" MM="04" DD="05"/>
      // endDate is not required
    </startDate>
  </dateRange>
</ECHOInsertDate>
```

---

## ECHO Last Update

Use **ECHOLastUpdate** to specify a search for collections or granules based on the date of their last update within ECHO. You can only specify a **dateRange**. You cannot use the negated condition for this search.

---

```
<ECHOLastUpdate>
  <dateRange>
    <startDate>
      <Date YYYY="2001" MM="01" DD="01"/>
    </startDate>
    <stopDate>
      <Date YYYY="2001" MM="06" DD="01"/>
    </stopDate>
  </dateRange>
</ECHOLastUpdate>
```

---

## Sensor Name

Use **sensorName** to search for collections or granules based on the name of the sensor. You can search for a known value in a **value**, a list of known values in a **list**, or a text pattern in a **textPattern**.

You can specify the relationship between each value with the Boolean "AND" or "OR". The "AND" relationship means that results must match every value specified. The "OR" relationship means that results must match either or both of the values specified. "OR" is the default setting for searching with multiple values.

---

```
<granuleCondition>
  <sensorName operator="AND">
    <list>
      <value>'human observer'</value>
      <value>'rain gauge'</value>
    </list>
  </sensorName>
</granuleCondition>
```

---



## Source Name

Use **sourceName** to search for collections or granules based on the name of the source. You may specify the name using a **value**, a list of known values in a **list**, or a text pattern in a **textPattern**. You can specify the relationship between each value with the Boolean "AND" or "OR". The "AND" relationship means that results must match every value specified. The "OR" relationship means that results must match either or both of the values specified. "OR" is the default setting for searching with multiple values

```
<granuleCondition>
  <sourceName operator="AND">
    <list>
      <value>'CV-580'</value>
      <value>'C-130'</value>
    </list>
  </sourceName>
</granuleCondition>
```

## Spatial

Use spatial constraints to restrict the search within a spatial extent of the earth. You cannot use the negated condition for this search and cannot use spatial constraints in conjunction with the **globalGranulesOnly** condition.

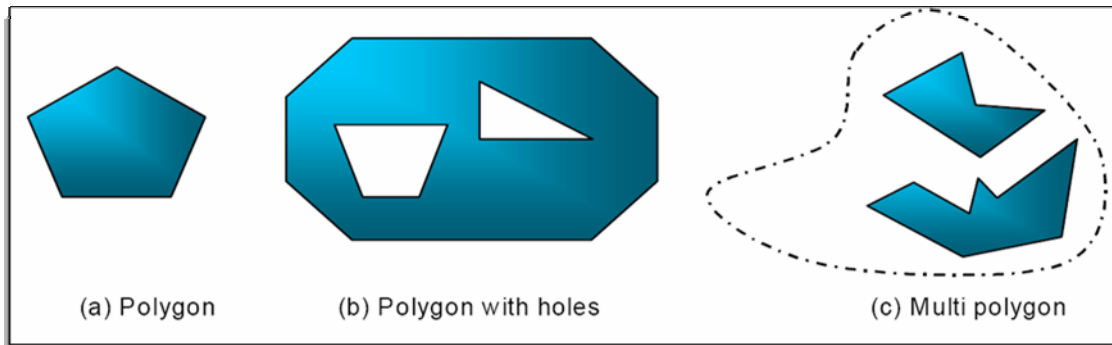
Specify the spatial extent using the OpenGIS Consortium's (OGC) Geography Markup Language (GML) elements **Polygon**, **MultiPolygon** or **LString** found in the GeometryCollection.dtd or by using the ECHO elements **IIMSPolygon**, **IIMSMultiPolygon**, or **IIMSLine**. *When using GML elements, you must specify the point with longitude first followed by latitude.*

You can also specify a **TwoDCoordinate** element along with a spatial element in a spatial constraint. This is useful when you want to find spatially coincident granules between collections that support **TwoDCoordinate** searches and those that do not. When you use **TwoDCoordinate** together with a spatial element, ECHO will return all granules that satisfy either the **TwoDCoordinate** constraint or the spatial element, or both. You can only use **TwoDCoordinate** in a spatial constraint for granule (inventory) queries, not collection queries.

You can also use **TwoDCoordinate** outside of a spatial constraint. In this case, list **TwoDCoordinate** inside a **granuleCondition**. ECHO will return results that satisfy the **TwoDCoordinate** constraint and all other conditions (including any spatial conditions if they are present).

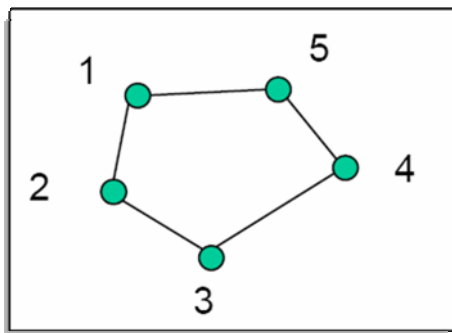
A **Polygon** consists of one or more rings. Some rings can be embedded within others that define the interior of the polygon. *The ECHO system can handle polygons with only one external ring and multiple internal rings.*

Figure 2: Allowed Geometries for Spatial Data and Query Windows



In order to define a ring, the last point in the ring should be the same as the first point in the ring. You must specify the points in each polygonal ring in counterclockwise order.

Figure 3: Order of Points in a Polygonal Ring



To specify multiple outer rings, use the **MultiPolygon** or **IIMSMultiPolygon** element, with each outer ring in a separate **Polygon**. There should be no overlap between any of the outer rings defined.

Use **IIMSLine** or **LString** to specify a line on the Earth. You must specify the points in the line from one end point to the other. You can also use **IIMSPoint** to specify a single point on the Earth. Note that although **IIMSPoint** is used to describe corner points in a polygon or end points in a line, it also describes a specific searchable geometry.

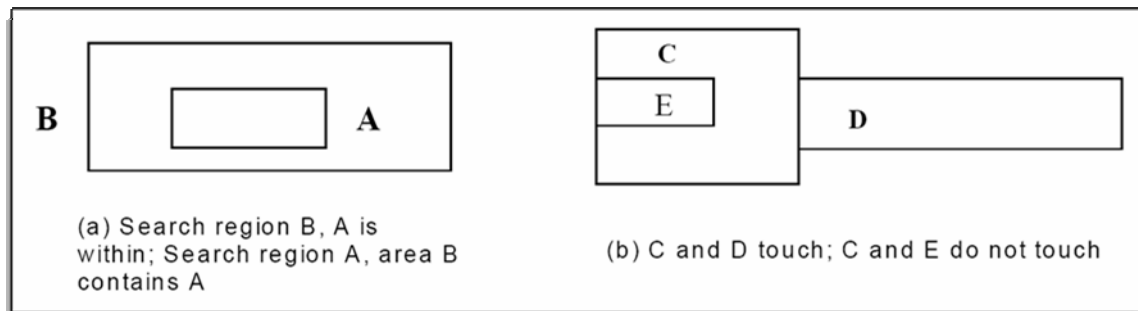
The table below describes Spatial Operators available in the ECHO System, regarding the relationship between two objects.

Table 5: Spatial Operators

| Spatial Operator | Description  |
|------------------|--|
| EQUAL            | They have the same boundary and interior.                |
| TOUCH            | The boundaries intersect but the interiors do not.       |
| WITHIN           | Interior and boundary of the first object are completely |

| Spatial Operator | Description   |
|------------------|---|
|                  | contained in the interior of the second.  |
| CONTAINS         | The interior and boundary of the second object are completely contained in the interior of the first. |
| RELATE           | The objects are non-disjoint, that is, their bodies and/or boundaries intersect.                      |

Figure 4: Examples of Spatial Regions



Define the query so that the user-specified spatial extent is the second object in the query.

- Use **CONTAINS** to retrieve granules/collections that contain the specified polygon.
- Use **WITHIN** to retrieve granules/collections that are within the specified spatial extent.
- Use **RELATE** to retrieve granules/collections that overlap in some way. **RELATE** is the default operator.

ECHO supports multiple spatial representations (or SpatialTypes). Spatial data of different spatial representations are described differently and hence are stored and queried differently. The spatial types that ECHO currently supports are described below.

The **SpatialType** tag is an optional field in AQL used to specify the spatial type. If the user opts not to specify this field, ECHO performs a default search. By default, ECHO performs the search on **NORMAL** (Cartesian and Geodetic), **CONSTANT\_COVERAGE** and **ORBIT**. **GLOBAL** granules are not returned in the default case.

Code Listing 33: Spatial Search Element

---

```
<spatial operator="RELATE">
  <Polygon>
    <LRing>
      <CList>
        -120, -30, -100, -60, 5, -90,
        -120, -60, 160, 5, 160, 60, 120, 85,
        5, 85, -120, 30, -120, -30
      </CList>
    </LRing>
    <LRing>
      <CList>
        80, 20, 80, 60, 20, 60, 20, 20, 80, 20
      </CList>
    </LRing>
  </Polygon>
  <SpatialType>
    <list>
      <value>NORMAL</value>
      <value>ORBIT</value>
      <value>GLOBAL</value>
    </list>
  </SpatialType>
</spatial>
```

---

### Spatial Type: NORMAL

Conventional spatial data are described as a certain shape such as a polygon, a multi-polygon, or a line. **NORMAL** is the spatial type for this type of data. Spatial data of this type are stored in the database as Oracle objects to record the shape, spatial locations and coordinate system used, that is, Cartesian or Geodetic.

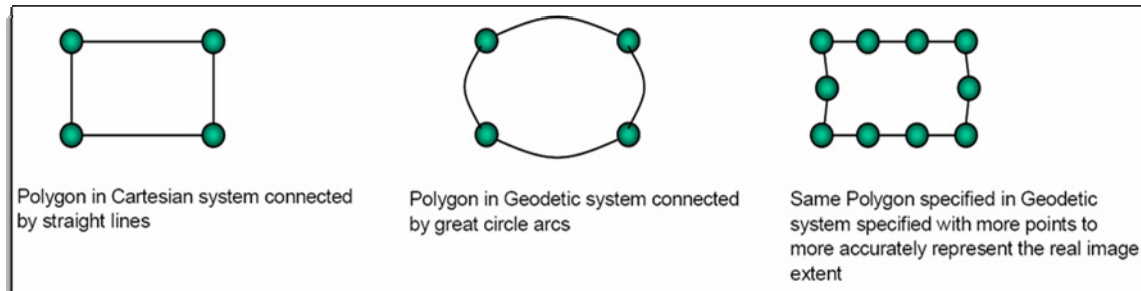
The Oracle search for spatial data has different restrictions for different coordinate systems.

- *For the Cartesian system*, Oracle allows the coverage to be as large as the whole earth but does not allow data that covers the pole or crosses the date line, so the search must conform to these constraints.
- *For the Geodetic system*, Oracle allows the data to cross the date line and the poles, but the total area of a single polygon cannot exceed half the earth, so the search must conform to these constraints.

Oracle does not validate the window covered by the spatial search before executing the query. If the window is invalid, you may receive incorrect results but will not receive an

Oracle error. To reduce the effect of this issue on clients, ECHO itself validates the spatial window to conform to the Geodetic coordinate system. When you query Cartesian system data, it is your responsibility to make sure that your search conforms to the extra restrictions. In addition, even though the Cartesian system allows the query window to be as large as the whole earth, ECHO will restrict the window to be no larger than half the area of the earth. Smaller sized spatial search windows result in faster response times.

**Figure 5: Polygon Area Defined Depends on the Number of Points and the Coordinate System Used**

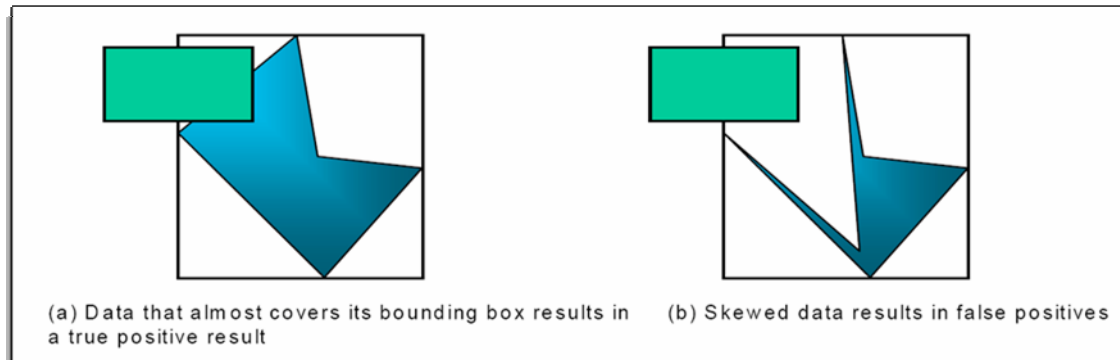


The actual area on the earth being queried depends on the coordinate system used by the provider and the number of points in the query window. When searching data from providers using the Cartesian coordinate system, the search polygon is converted to a Cartesian polygon with points connected using straight lines. When searching data from providers using the Geodetic coordinate system, the search polygon is converted to a Geodetic polygon with points connected using great circle arcs. This may result in slightly different results for data from providers using the different coordinate systems. To ensure better accuracy, you should represent the search window with more than just the corner points.

Oracle Spatial search is a two-step process. The first step filters the data by querying the spatial window against the bounding box, where applicable, of the spatial data. The second step searches within the results of the first step, with a slower, more accurate polygonal geometry search for exact results. The filtering provided during the first step allows the second step of the search to run faster.

Skewed polygonal windows or data with lots of wasted space in the bounding box will result in more false positives passing through the fast filter. Data orientation with respect to its bounding rectangle may play a role in the response time of a query, as shown below:

Figure 6: Example Data Orientations



### Spatial Type: GLOBAL

A special case is if the spatial coverage of the data is the entire earth. The data does not come with any specific spatial value, but rather a flag to denote its spatial as global. Currently ECHO only supports global spatial searches on granules.

### Spatial Type: ORBIT

Spatial data are orbit-based and generated from satellites running along orbits. Because an orbit travels at a fixed direction with a fixed declination angle and the covered area is a stripe of fixed width, the spatial coverage of a granule can be derived from the latitude where the granule starts and ends and the crossing longitude on the equator where its orbit originally starts. Thus, the orbit-based spatial data are not stored as shape objects but as the original data. ECHO uses an algorithm called “backtrack” to perform searches on orbit-based data. Spatial coverage can be as large as the entire earth surface including polar area. ECHO does simple validation against the spatial window to make sure it does not exceed the scope of the earth (the latitude is within the range  $-90$  to  $90$  and the longitude is within the range  $-180$  to  $180$ ).

There are some limitations in the current implementation. First, ECHO only supports searching on single-orbit granules. Secondly, the only supported spatial operator is **RELATE**.

This example shows a multi-polygon spatial query.

**Code Listing 34: A Multi-Polygon Spatial Query**

---

```
<spatial operator='RELATE'>
  <IIMSMultiPolygon>
    <IIMSPolygon>
      <IIMSLRing>
        <IIMSPoint lat="85" long="-50"/>
        <IIMSPoint lat="70" long="-60"/>
        <IIMSPoint lat="60" long="-50"/>
        <IIMSPoint lat="70" long="-40"/>
        <IIMSPoint lat="85" long="-50"/>
      </IIMSLRing>
    </IIMSPolygon>
    <IIMSPolygon>
      <IIMSLRing>
        <IIMSPoint lat="-85" long="-50"/>
        <IIMSPoint lat="-70" long="-40"/>
        <IIMSPoint lat="-60" long="-50"/>
        <IIMSPoint lat="-70" long="-60"/>
        <IIMSPoint lat="-85" long="-50"/>
      </IIMSLRing>
    </IIMSPolygon>
  </IIMSMultiPolygon>
</spatial>
```

---

This example shows a single point spatial query.

**Code Listing 35: A Single Point Spatial Query**

---

```
<spatial operator='RELATE'>
  <IIMSPoint long='-75.0' lat='35.0'/>
</spatial>
```

---

This example shows a spatial query using a line defined by GML elements.

**Code Listing 36: A Spatial Query Using a Line Defined by GML Elements**

---

```
<spatial operator='EQUAL'>
  <LString>
    <CList>70, 70, 80, 80, 90, 90</CList>
  </LString>
</spatial>
```

---

## Temporal

Use temporal constraints to specify the temporal extent of the data. The dates are stored as Gregorian dates, and the times are stored in GMT. This search allows only dates in YYYY-DD-MM format. You can specify a range of time represented by beginning and ending dates or a periodic temporal range specified by start date and end date. For a periodic temporal range, you can specify the start and end day of each year. You should use integers between 1 and 365 to specify the start and end dates, with start day  $\leq$  end day. You can specify the end day as 366, but if the temporal range contains a non-leap year, then an error will be generated.

This example shows a search for granule/collections whose temporal range overlaps with the time between May 12, 1990, and June 13, 1992.

### Code Listing 37: Temporal Search Example

---

```
<temporal>
  <startDate><Date YYYY="1990" MM="5" DD="12"/></startDate>
  <stopDate><Date YYYY="1992" MM="6" DD="13"/></stopDate>
</temporal>
```

---

This example shows a search for granules/collections whose temporal range overlaps with all of these time ranges:

May 16 - July 14, 1990

January 5 - March 5, 1991

January 5 - February 10, 1992

### Code Listing 38: Temporal Example with Different Day Notations

---

```
<temporal>
  <startDate><Date YYYY="1990" MM="5" DD="12"/></startDate>
  <stopDate><Date YYYY="1992" MM="2" DD="10"/></stopDate>
  <startDay value="5"/>
  <endDay value="60"/>
</temporal>
```

---



## Orderable Items

Use **orderable** to specify a search for granules or collections that are available for ordering.

### Code Listing 39: Using the Orderable Condition

---

```
<orderable/>
```

---

## Version ID

Use **versionId** to search for collections or granules based on the version identifier of the data collection. You can search for a known value in a **value**, a list of known values in a **list**, or a text pattern in a **textPattern**. In ECHO 9.0, versioned is stored as a string so that it can look like 3,0033,0 ....

### Code Listing 40: Version ID in a Collection Condition

---

```
<versionId>
  <value>'0'</value>
</versionId>
```

---

### Code Listing 41: Version ID in a Granule Condition with a Pattern

---

```
<versionId>
<textPattern operator="LIKE">'3.%'</textPattern>
</versionId>
```

---

## Additional Attributes

Use **additionalAttributes** to search for collections or granules based on one or many groups of the additional attributes by specifying the additional attribute name/value pair for an exact match. This function includes searching on multiple additional attributes. You can specify the relationship between groups with either "AND" or "OR". The "AND" relationship means that result granules must match across every group. The "OR" relationship means that result granules match with any of the specified groups. "OR" is the default setting.

Within each additional attribute, you must specify an **additionalAttributeName** in *single quotation marks (that is, apostrophes)* for an exact match. You may specify the **additionalAttributeValue** using **value**, **list**, **textPattern**, **range**, or **dateRange**.

This example shows the "AND" relationship between two additional attribute groups. You can use this sample search with either **granuleCondition** or **collectionCondition**.

#### Code Listing 42: Additional Attributes Example with an AND Relationship

---

```
<additionalAttributes operator='AND'>
  <additionalAttribute>
    <additionalAttributeName>
      'AveragedFocalPlane1Temperature'
    </additionalAttributeName>
    <additionalAttributeValue>
      <range lower='169' upper='170' />
    </additionalAttributeValue>
  </additionalAttribute>
  <additionalAttribute>
    <additionalAttributeName>
      'AveragedFocalPlane2Temperature'
    </additionalAttributeName>
    <additionalAttributeValue>
      <range lower='269' upper='270' />
    </additionalAttributeValue>
  </additionalAttribute>
</additionalAttributes>
```

---

This example shows the "OR" relationship between two additional attribute groups.

#### Code Listing 43: Additional Attribute Example with an OR Relationship

---

```
<additionalAttributes operator='OR'>
  <additionalAttribute>
    <additionalAttributeName>
      'AveragedFocalPlane1Temperature'
    </additionalAttributeName>
    <additionalAttributeValue>
      <range lower='169' upper='170' />
    </additionalAttributeValue>
  </additionalAttribute>
  <additionalAttribute>
    <additionalAttributeName>
      'AveragedFocalPlane2Temperature'
    </additionalAttributeName>
    <additionalAttributeValue>
      <range lower='269' upper='270' />
    </additionalAttributeValue>
  </additionalAttribute>
</additionalAttributes>
```

---

## Search Elements for Discovery Queries

This section describes elements used specifically for collection (Discovery) searches.

### Parameter

Use **parameter** to specify the geophysical terms associated with a collection. If you specify a list or a single value, then the search looks for an exact match between the string specified and the data stored. For example, if the search value is 'Imagery' and the data stored has the value 'Satellite Imagery', the data will not be returned since there was no exact match. In this instance, it is more useful to use the **textPattern** element for the search.

#### Code Listing 44: Sample Parameter Search

---

```
<collectionCondition>
  <parameter>
    <textPattern operator="LIKE">%Imagery%</textPattern>
  </parameter>
</collectionCondition>
```

---

### Processing Level

Use **processingLevel** to specify the level at which the data (collection/granule) has been processed, for example, level 0, 1, 1A, 1B, 2, 3, 4. The search can specify one or more processing levels. The values must be single-quoted strings.

This example shows a search for all collections at processing level 1, 1A, or 1B.

#### Code Listing 45: Processing Level Search Element

---

```
<collectionCondition>
  <processingLevel>
    <list>
      <value>'1'</value>
      <value>'1A'</value>
      <value>'1B'</value>
    </list>
  </processingLevel>
</collectionCondition>
```

---

This example shows a search for all collections at processing level 1, 1A, or 1B—any value starting with the digit 1. This also returns results for collections with processing levels 13, 14, etc.

Code Listing 46: Processing Level Element with Text Pattern

---

```
<collectionCondition>
  <processingLevel>
    <textPattern operator="LIKE">'1_</textPattern>
  </processingLevel>
</collectionCondition>
```

---

### Additional Attribute Name

Use **additionalAttributeName** to search for collections based on the names of the additional attributes in the collections. You can search for a known value in a **value**, a list of known values in a **list**, or a text pattern in a **textPattern**.

This example shows a search for data with an **additionalAttributeName** value of 1 or 3.

Code Listing 47: Additional Attribute Search Element

---

```
<collectionCondition>
  <additionalAttributeName>
    <list>
      <value>'value_1'</value>
      <value>'value_3'</value>
    </list>
  </additionalAttributeName>
</collectionCondition>
```

---

### Short Name

Use **shortName** to specify a search for the short name of collections, which may or may not be unique for a particular provider.

This example shows a search for all collections that have shortNames starting with 'BOREAS.'

Code Listing 48: Short Name Search Element

---

```
<collectionCondition>
  <shortName>
    <textPattern>'BOREAS%'</textPattern>
  </shortName>
</collectionCondition>
```

---

## Spatial Keywords

Use **spatialKeywords** to specify a word or phrase that summarizes the spatial region covered by the collection. A collection may cover several regions.

This example shows a search for all collections that cover Africa or Bermuda or the Indian Ocean.

Code Listing 49: Spatial Keywords Search Element

---

```
<collectionCondition>
  <spatialKeywords>
    <list>
      <value>'Africa'</value>
      <value>'Bermuda'</value>
      <value>'Indian Ocean'</value>
    </list>
  </spatialKeywords>
</collectionCondition>
```

---

This example shows a search for all collections that cover some region(s) of the Americas.

Code Listing 50: Spatial Keywords Element with Text Pattern

---

```
<collectionCondition>
  <spatialKeywords>
    <textPattern operator="LIKE">'%america%'</textPattern>
  </spatialKeywords>
</collectionCondition>
```

---

## Temporal Keywords

Use **temporalKeywords** to specify a word or phrase that summarizes the temporal characteristics referenced in the collection.

This example shows a search for all collections that have accumulated data during the spring or fall.

Code Listing 51: Temporal Keywords Element

---

```
<collectionCondition>
  <temporalKeywords>
    <list>
      <value>'spring'</value>
      <value>'fall'</value>
    </list>
  </temporalKeywords>
</collectionCondition>
```

---

## Archive Center

Use **archiveCenter** to search for collections based on the center where a collection is archived. You may search for a known value in a **value**, a list of known values in a **list**, or a text pattern in a **textPattern**.

Code Listing 52: Archive Center Search Element

---

```
<collectionCondition>
  <archiveCenter>
    <value>'ORNL_DAAC'</value>
  </archiveCenter>
</collectionCondition>
```

---

## Search Elements for Inventory Queries

This section describes elements used specifically for granule (Inventory) searches.

### Granules with Browse Data

Use **browseOnly** to specify that you want only granules that have browse data available. You cannot use the negated condition for this search.

Code Listing 53: Browse-only Search Element

---

```
<granuleCondition>
  <browseOnly/>
</granuleCondition>
```

---

### Percentage of Cloud Cover

Use **cloudCover** to specify a range of percentage of scene cloud coverage in a granule. For the value in each attribute of the range, you should use a floating number between 0 and 100 that specifies the range in percentage of cloud coverage in a granule. You cannot use the negated condition for this search.

This example shows a search for only those granules that have cloud cover between 10% and 20%.

Code Listing 54: Cloud Cover Search Element

---

```
<granuleCondition>
  <cloudCover>
    <range lower="10" upper="20"/>
  </cloudCover>
</granuleCondition>
```

---

### Day, Night, or Both Granules

Use **dayNightFlag** to specify a search for granules that are gathered during daylight only, nighttime only, or both. If you do not specify this criterion, then your results will include

granules gathered at any and all times of day. You cannot use the negated condition for this search.

This example shows a search for granules gathered at least partly during daylight.

**Code Listing 55: Search Element for Granules Gathered During Daylight Only**

---

```
<granuleCondition>
  <dayNightFlag value="DAY"/>
</granuleCondition>
```

---

This example shows a search for granules gathered at least partly during the night.

**Code Listing 56: Search Element for Granules Gathered During the Nighttime Only**

---

```
<granuleCondition>
  <dayNightFlag value="NIGHT"/>
</granuleCondition>
```

---

This example shows a search for granules gathered partly at night and partly during the daylight.

**Code Listing 57: Search Element for Granules Gathered During Daylight, Nighttime, or Both**

---

```
<granuleCondition>
  <dayNightFlag value="Both"/>
</granuleCondition>
```

---

## Only Global Granules

Use **globalGranulesOnly** to search only for Global granules (granules that span the whole earth). Do not specify both spatial criteria and **globalGranulesOnly** as granule conditions. Only one of these criteria is allowed. You cannot use the negated condition for this search.

**Code Listing 58: Global Granules Only Search Element**

---

```
<granuleCondition>
  <globalGranulesOnly/>
</granuleCondition>
```

---

## ECHO Granule IDs

Use **ECHOGranuleID** to specify an inventory search for specific granules. This search does not need to specify any spatial or temporal constraint. You can use this as a second stage query when you know the list of granules you are interested in from a previous query. The search is then limited to granules in the list. The list may contain granules from different data sets. ECHO performs the search on `ECHO_GRANULE_ID`, an ECHO-wide unique ID. You cannot use the negated condition for this search. *You must enclose the names of the granules in single quotation marks (that is, apostrophes).*

#### Code Listing 59: Granule ID Search Element

---

```
<granuleCondition>
  <ECHOGranuleID>
    <list>
      <value>'G1984-ORNL_DAAC'</value>
    </list>
  </ECHOGranuleID>
</granuleCondition>
```

---

### Granule UR

Use **GranuleUR** to specify an inventory search for specific granules. Like granule ID, this criterion can be used as a secondary stage query. Instead of using the ECHO-specific ID for the granules, this criterion allows you to use the provider-given name for the granules. Provider-given names are more meaningful and easier to remember. Since the granule names given by the provider are not guaranteed to be unique within ECHO, if a granule with the same name exists for more than one **dataCenterId** specified in the query, all will be returned in the results. You cannot use the negated condition for this search. *You must enclose the names of the granules in single quotation marks (that is, apostrophes).*

#### Code Listing 60: Granule UR Search Element

---

```
<granuleCondition>
  <GranuleUR>
    <list>
      <value>'ACCP_CANOPYCHEM.df_can_calc.dat'</value>
      <value>'ANGLIA_10YRCLIMATE.cfrs1120.zip'</value>
    </list>
  </GranuleUR>
</granuleCondition>
```

---

### Two-Dimensional Coordinate System

Use **TwoDCoordinateSystem** as an alternative to specifying a spatial or temporal range. For some collections, granules can be located by searching a range of values on a two dimensional grid, (for example, Path/Row for Landsat, X/Y for MODIS Grid data). You cannot use the negated condition for this search (that is, if the parent element has the attribute `negated='y'`, the NOT operator will not be added and the condition remains positive).

The **TwoDCoordinateSystem** criterion needs three elements:

- the **TwoDCoordinateSystemName**
- the **Coordinate1** range
- the **Coordinate2** range

**Coordinate1** and **Coordinate2** may also take a single value. The query returns all granules that match your specified **TwoDCoordinateSystemName** and overlap with the



**Coordinate1** and **Coordinate2** ranges. **Coordinate1** and **Coordinate2** ranges must have both lower and upper attributes specified.

This example shows a search for a range value.

**Code Listing 61: TwoDCoordinateSystem Search Element**

---

```
<granuleCondition>
  <TwoDCoordinateSystem>
    <TwoDCoordinateSystemName>
      <value>'WRS2'</value>
    </TwoDCoordinateSystemName>
    <Coordinate1><range lower='15' upper='20' /></Coordinate1>
    <Coordinate2><range lower='33' upper='42' /></Coordinate2>
  </ TwoDCoordinateSystem>
</granuleCondition>
```

---

This example shows a search for a single value.

**Code Listing 62: TwoDCoordinateSystem Search Element with a Single Value**

---

```
<granuleCondition>
  <TwoDCoordinateSystem>
    <TwoDCoordinateSystemName>
      <value>'WRS2'</value>
    </TwoDCoordinateSystemName>
    <coordinate1><value>15<value /></coordinate1>
    <coordinate2><range lower='33' upper='42' /></coordinate2>
  </ TwoDCoordinateSystem >
</granuleCondition>
```

---

## ProducerGranuleID

Use **ProducerGranuleID** to specify an inventory search for specific granules. **ProducerGranuleID** represents the ID assigned to data by the Science team that produced the data. Instead of using the ECHO-specific IDs for the granules, this criterion allows the user to use the producer's given name for the granules. Granule names are available to the Science teams and are easier to search on. Since the granule names given by the producer are not guaranteed to be unique within ECHO, if a granule with the same name exists for more than one **dataCenterId** specified in the query, all will be returned in the results. You cannot use the negated condition for this search. For example, if the parent **granuleCondition** element has the attribute `negated='y'`, the NOT operator will not be added and the condition remains positive. *You must enclose the names of the granules in single quotation marks (that is, apostrophes).*

#### Code Listing 63: ProducerGranuleID Search Element

---

```
<granuleCondition>
  <ProducerGranuleID>
    <list>
      <value>'MOD03.A2000107.0930.003.2002056052717.hdf'</value>
      <value>'MOD01.A2000107.0740.003.2002056051321.hdf'</value>
    </list>
  </ProducerGranuleID>
</granuleCondition>
```

---

### PGE Name

Use **PGEName** to search for granules based on the name of the product generation executive (PGE). You can search for a known value in a **value** element, a list of known values in a **list**, or a text pattern in a **textPattern**.

#### Code Listing 64: PGE Name Search Element

---

```
<granuleCondition>
  <PGEName>
    <value>'test_PGEName'</value>
  </PGEName>
</granuleCondition>
```

---

### PGE Version

Use **PGEVersion** to search for granules based on the version of PGE that originally produced the granule. You can search for a known value in a **value** element, a list of known values in a **list**, or a text pattern in a **textPattern**.

#### Code Listing 65: PGE Version Search Element

---

```
<granuleCondition>
  <PGEVersion>
    <value>'3.0.0'</value>
  </PGEVersion>
</granuleCondition>
```

---

## Collection Short Name

Use **collectionShortName** to search for granules based on the short name of the collection they belong to. You can search for a known value in a **value** element, a list of known values in a **list**, or a text pattern in a **textPattern**.

Code Listing 66: Collection Short Name Search Element

```
<granuleCondition>
  <collectionShortName>
    <value>'MOD03'</value>
  </collectionShortName>
</granuleCondition>
```

## Measured Parameters

Use **measuredParameters** to search for granules based on one or more groups of measured science parameters. You can specify the relationship between groups of parameters with either AND or OR. An AND relationship means that result granules must match across all groups. An OR relationship means that result granules can match any of the groups. OR is the default setting.

Within each measured parameter, you must specify a **measuredParameterName** in single quotation marks (that is, apostrophes) for exact matching. The detailed measured parameters can be **value**, **list**, and **textPattern** type, or **range** and **dateRange** type.

This example shows a search that specifies an "AND" relationship between two measured parameter groups.

Code Listing 67: Measured Parameter Search Element

```
<granuleCondition>
  <measuredParameters operator="AND">
    <measuredParameter>
      <measuredParameterName>
        'EV_1KM_RefSB'
      </measuredParameterName>
      <operationalQualityFlag>
        <value>'Passed'</value>
      </operationalQualityFlag>
      <QAPercentOutOfBoundsData>
        <range lower='30' upper='40' />
      </QAPercentOutOfBoundsData>
    </measuredParameter>
    <measuredParameter>
      <measuredParameterName>
        'EV_1KM_RefSB'
      </measuredParameterName>
      <operationalQualityFlag>
        <value>'Passed'</value>
      </operationalQualityFlag>
      <QAPercentOutOfBoundsData>
        <range lower='39' upper='50' />
      </QAPercentOutOfBoundsData>
    </measuredParameter>
  </measuredParameters>
</granuleCondition>
```

---

```
    </measuredParameter>
  </measuredParameters>
</granuleCondition>
```

---

This example shows a search that specifies an "OR" relationship between two measured parameter groups.

**Code Listing 68: Measured Parameter Search Element**

---

```
<granuleCondition>
  <measuredParameters operator="OR">
    <measuredParameter>
      <measuredParameterName>
        'EV_1KM_RefSB'
      </measuredParameterName>
      <operationalQualityFlag>
        <value>'Passed'</value>
      </operationalQualityFlag>
      <QAPercentOutOfBoundsData>
        <range lower='30' upper='40' />
      </QAPercentOutOfBoundsData>
    </measuredParameter>
    <measuredParameter>
      <measuredParameterName>
        'EV_1KM_RefSB'
      </measuredParameterName>
      <operationalQualityFlag>
        <value>'Passed'</value>
      </operationalQualityFlag>
      <QAPercentOutOfBoundsData>
        <range lower='39' upper='50' />
      </QAPercentOutOfBoundsData>
    </measuredParameter>
  </measuredParameters>
</granuleCondition>
```

---

### ProviderInsertDate

Use **providerInsertDate** to search for granules that are within a desired date-time range. **StartDate** is a required field, and **StopDate** is optional. The **Date** format is:

---

```
<Date YYYY="2001" MM="01" DD="01" HH="00" MI="00" SS="00"/>
```

---

Where YYYY is year; MM is month, DD is day of the month; HH is hour; MI is minute, and SS is second.

This example shows a search using **providerInsertDate**.

**Code Listing 69: Provider Insert Date Search Element**

---

```
<granuleCondition>
  <providerInsertDate>
    <dateRange>
      <startDate>
        <Date YYYY="2001" MM="01" DD="01" HH="00" MI="00"
SS="00"/>
      </startDate>
      <stopDate>
        <Date YYYY="2003" MM="12" DD="01" HH="23" MI="59"
SS="59"/>
      </stopDate>
    </dateRange>
  </providerInsertDate>
</granuleCondition>
```

---

### ProviderProductionDate

Use **ProviderProductionDate** to search for granules that are within a desired date-time range. **StartDate** is a required field, and **StopDate** is optional. The **Date** format is:

---

```
<Date YYYY="2001" MM="01" DD="01" HH="00" MI="00" SS="00"/>
```

---

Where YYYY is year; MM is month, DD is day of the month; HH is hour; MI is minute, and SS is second.

This example shows a search of the Data Provider's production date search element.

**Code Listing 70: Provider Production Date Search Element**

---

```
<granuleCondition>
  <providerProductionDate>
    <dateRange>
      <startDate>
        <Date YYYY="2001" MM="01" DD="01" HH="00" MI="00"
SS="00"/>
      </startDate>
      <stopDate>
        <Date YYYY="2003" MM="12" DD="01" HH="23" MI="59"
SS="59"/>
      </stopDate>
    </dateRange>
  </providerProductionDate>
</granuleCondition>
```

---

# Miscellaneous Elements

Use the following elements to search for other elements using AQL.

## List

Use **List** to specify a list of values. The values may be numbers, dates, or strings. Each individual value must appear as text of the value element. If you specify a list of strings, you must use a single-quoted string with no wild-card characters. ECHO searches the parent element using Oracle's IN operator for an exact match with any value in the list.

In general, interpretation of the list will be dependent on the parent element in which it appears.

Code Listing 71: The List Element

```
<list>
  <value>'Africa'</value>
  <value>'Bermuda'</value>
  <value>'Indian Ocean'</value>
</list>
```

## Text Pattern

Use **textPattern** to search using the Oracle operator “LIKE” to perform pattern matching. Because of pattern matching, exact specification of what is desired is not needed. The code listing below shows how to invoke the pattern matching. Note that you must specify the operator, and you must use a single-quoted string with or without wild-card characters.

The wild-card characters supported are '%' and '\_'. The character '%' is a placeholder that represents any number of characters. The character "\_" is a placeholder that represents a single character.

Table 6: AQL Wildcard Use

| Wildcard pattern | Interpretation                               |
|------------------|--|
| '%JOHN%'         | Strings containing 'JOHN' as a sub-string    |
| 'MARY%'          | Strings beginning with the characters 'MARY' |
| '%BOB'           | Strings ending in 'BOB'                      |
| 'D_VE'           | Strings like 'DAVE', 'DOVE', etc.            |

You can include the actual characters "%" or "\_" in the pattern by using the ESCAPE character "\. If the escape character appears in the pattern before the character "%" or "\_", then Oracle interprets this character literally in the pattern, rather than as a special pattern-matching character.

Table 7: Wildcard Examples with ESCAPE Character

| Wildcard pattern with ESCAPE character | Interpretation                            |
|--|---|
| 'JOHN\%THAN'                           | String 'JOHN%THAN'                        |
| 'JOHN%THAN'                            | Strings 'JOHNNATHAN', 'JOHNASDTHAN', etc. |
| 'JOHN\\MARY'                           | String 'JOHN\MARY'                        |

If the parent element (**collectionCondition** or **granuleCondition**) has the attribute negated with value 'Y', then the search will look for a string NOT like the pattern specified.

#### Code Listing 72: Text Pattern Element

```
<spatialKeywords>
  <textPattern operator="LIKE"
    caseInsensitive="Y">'america%'</textPattern>
</spatialKeywords>
```

### Value

Use **value** to specify one value. Refer to each individual parent element for a detailed explanation. In general, if the element represents a string pattern, the search will be case sensitive unless the **caseInsensitive** attribute is set to Y.

```
<value>12.6</value>
```

or

```
<value caseInsensitive="Y">'Asx'</value>
```

### Date

Use **Date** to specify a date value.

```
<Date YYYY="2000" MM="05" DD="03"/>
```

or

```
<Date YYYY="2000" MM="10" DD="09" HH="21" MI="04" SS="32"/>
```

### Date Range

Use **dateRange** to specify a range of time. The range can include just a **startDate**, just a **stopDate** or both. If you only include a **startDate**, ECHO searches for data from that

time forward with no stop date. If you only include a **stopDate**, ECHO searches for data from the current time up until the stop date. If you include both a **startDate** and a **stopDate**, then ECHO searches for data falling between those two dates, for example, between January 1, 2005 and January 31, 2005.

---

```
<dateRange>
  <startDate><Date YYYY="2005" MM="01" DD="01"/></startDate>
  <stopDate><Date YYYY="2005" MM="01" DD="31"/></stopDate>
</dateRange>
```

---

or

---

```
<dateRange>
  <stopDate>
    <Date YYYY="2000" MM="10" DD="09" HH="04" MI="12" SS="34"/>
  </stopDate>
</dateRange>
```

---

### DifEntryId

Use **DifEntryId** to find collections by their GCMD (Global Change Master Directory) DIF (Directory Interchange Format) Entry-ID. You can find the GCMD at the following URL: <http://gcmd.nasa.gov>.



## Chapter 6: Ordering Data Through ECHO

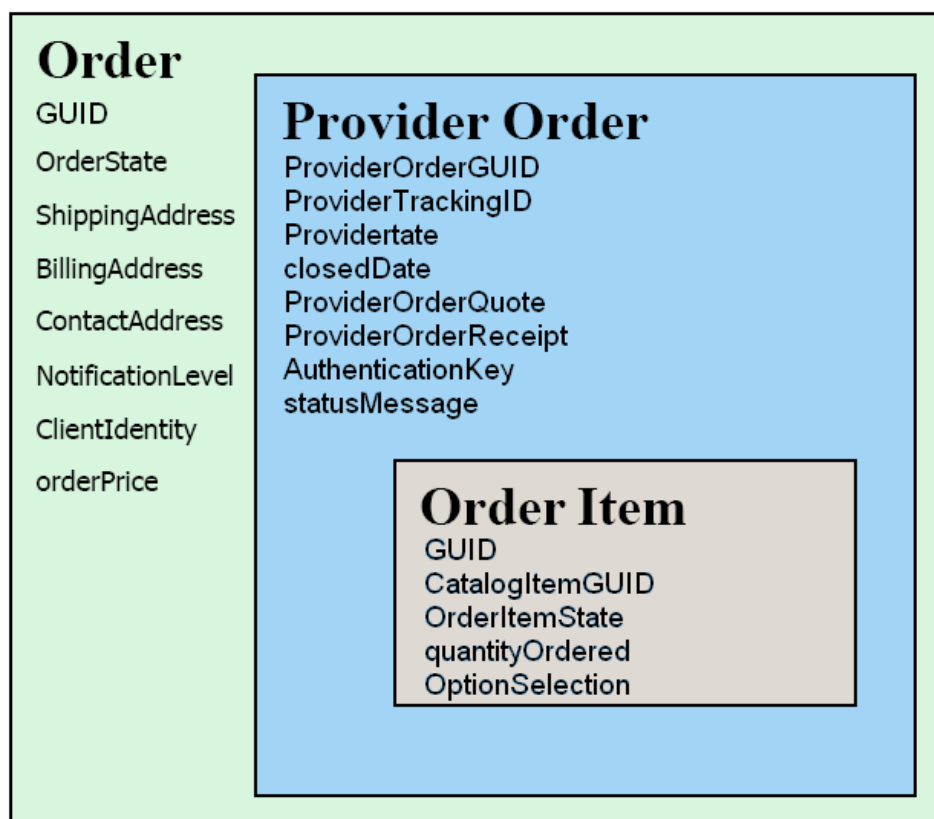
After identifying data of interest through catalog queries, you may place an order for that data if the data provider(s) allow it. *To create an order, it is critical that you understand the parts that make up an order.*

An order is a collection of **order items**. Each order item consists of:

- The GUID for the catalog item you wish to order
- The quantity of the item you wish to order
- Other options associated with the item

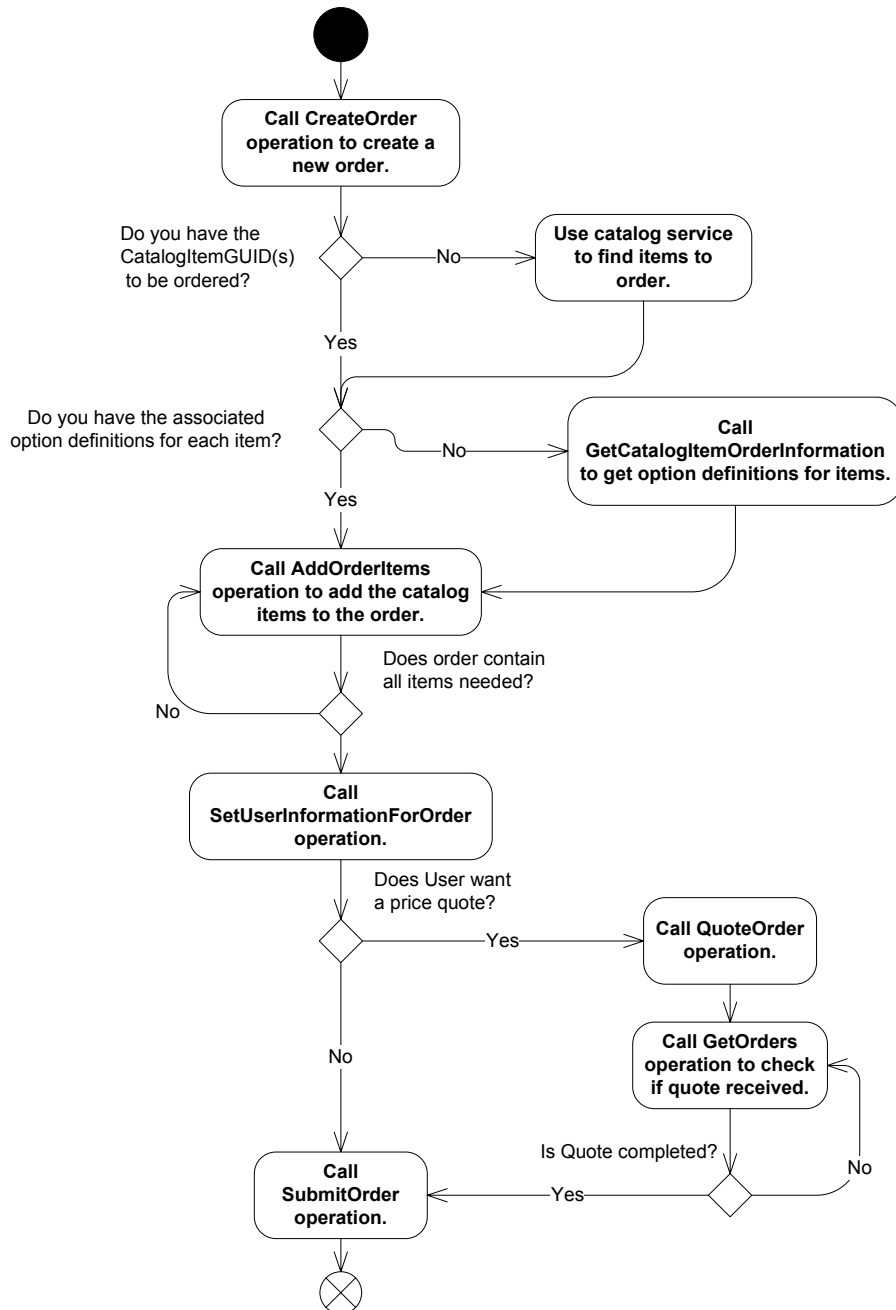
Many catalog items belonging to many different providers may comprise a single order. Each provider may have its own validation scheme and rules for creating, validating and submitting their particular catalog items. To ensure appropriate validation – as well as to demarcate a larger order for sending to each provider – orders are split up into smaller **provider orders**, sometimes called sub-orders. Each provider order includes all the order items in a particular order that belong to a specific provider. Each provider order has its own provider order GUID, a unique identifier comprised of the GUID of the provider and the GUID of the order that contains the catalog items for this provider. The following figure shows the structure of an order.

Figure 7: ECHO Order Structure



ECHO is designed to support a variety of providers and their various ordering needs, which brings some complexity to the process of order creation. The following figure illustrates the most commonly used and direct approach for creating and submitting an order.

Figure 9: The General Order Workflow



The most common way to obtain the catalog items needed for creating an order is through the **ExecuteQuery** operation found in the Catalog Service. You can also specify

how the results should be displayed. When doing so, ensure that the **CatalogItemID** tag is returned. The string value under this tag is the actual catalog item GUID that you can use to order items from ECHO.

## Order Options

Order Options use ECHO Forms (to [http://www.echo.nasa.gov/data\\_partners/data\\_tools9.shtml](http://www.echo.nasa.gov/data_partners/data_tools9.shtml) for more information) for providers to indicate additional information required from users in order to fulfill their orders, such as the media to put the ordered data on and the specific subset of the data to order. Providers assign Order Option definitions to catalog items. When you add an order item to an order that has required order options, it must contain an option selection that corresponds to one of the assigned option definitions for the catalog item you are ordering.

To list the possible options for each catalog item, use the **GetCatalogItemOrderInformation** operation on the **OrderManagementService**. The operation returns one object called **CatalogItemOrderInformation**. This contains a list of catalog items and a list of option definitions. Not every option definition in the list applies to every catalog item returned. Each catalog item is associated with zero or more option definition GUIDs, which will be included in the list returned in **CatalogItemOrderInformation**. Use only one selection from one of the definitions when you add a catalog item to the order. Use your discretion when choosing the definition to use for the order and the selection from within that definition.

*Note: Sometimes a catalog item may not be orderable. **GetCatalogItemOrderInformation** will return an exception if any catalog item is not orderable.*

*Note: Not all catalog items have associated definitions, in which case they do not need option selections when ordering. However, if there are any options that are required by a catalog item, you are required to fill them out before you can validate and/or submit the order, or else an error will be returned.*

The code listing below shows an example of getting the order options for two granules.

**Code Listing 73: Getting Order Options for a Catalog Item**

```
String token = "[token obtained from login]";

// The items we're interested in
String[] catalogItemGuids = new String[] { "G12345-PROV1",
"G23456-PROV1" };

// Get the service
OrderManagementServiceLocator locator = new
    OrderManagementServiceLocator();
OrderManagementServicePort orderService =
    locator.getOrderManagementServicePort();

// Get the order information
CatalogItemOrderInformation itemInfo =
    orderService.getCatalogItemOrderInformation(token,
catalogItemGuids);
```

## Creating an Order

Once you have collected the catalog item GUIDs and associated options, you can create an order by calling the **CreateOrder** operation on the **OrderManagementService**. Simply pass an array of item IDs to add to the order.

### Common Selection

If you are adding one or more items to an order at the same time, and their selections are the same, you can choose a common selection for the **CreateOrder**, **AddOrderItems**, **CreateAndSubmitOrder** and **UpdateOrderItems** operations that will be applied to all order items passed in without an option selection. This reduces the amount of data transmitted to ECHO and is less resource-intensive because ECHO will need to validate the common selection only once.

For each item in the array, you should set the following parameters:

**Table 8: Parameters on a Catalog Item**

| Parameter       | Description  |
|-----------------|--|
| ItemGuid        | The catalog item GUID from the item's metadata   |
| QuantityOrdered | The quantity of the item you want to order   |
| OptionSelection | Option selection that matches one of the option definitions associated with this item. If no required option definitions are associated with the item, this will be an empty list. |

The other attributes of Order Item should be empty when creating an order; ECHO will populate them automatically. ECHO will process the request and return a GUID identifying the newly created order. The code listing below shows an example of creating a simple order.

---

**Code Listing 74: Creating a Simple Order**

---

```
OrderItem[] orderItems = new OrderItem[2];
orderItems[0] =
    new OrderItem(null, null, "G12345-PROV1", null, (short)
2, null);
orderItems[1] =
    new OrderItem(null, null, "G23456-PROV1", null, (short)
2, null);
String orderGuid = orderService.createOrder(token,
orderItems,null);
```

---

### **Adding Order Items**

Once you have created an order, you may add other items to it using the **AddOrderItems** operation. This operation uses the GUID of the target order and a list of items to add. As with **CreateOrder**, the options other than those listed in the table above should be empty; ECHO will process the additions and return a list of GUIDs identifying these additions. The code listing below shows an example of adding items to an existing order.

---

**Code Listing 75: Adding Items to an Existing Order**

---

```
orderItems[0] =
    new OrderItem(null, null, "G34567-PROV1", null, (short)
2, null);
orderItems[1] =
    new OrderItem(null, null, "G45678-PROV1", null, (short)
2, null);
orderService.addOrderItems(token, orderGuid, orderItems,null);
```

---

*ECHO assigns each item in an order a GUID unique to that order. For example, adding the same granule to two different orders will produce two new GUIDs, one linking the granule to the first order and one linking the granule to the second order.*

## Updating Order Items

Once you have added an item to an order, you can modify the options or quantity by calling **UpdateOrderItems** and passing in the modified items. *In this case, you must fill in the GUID of each item since ECHO needs to find the original item in your order to update it.* To ensure you are updating the appropriate item with the correct information, follow these steps:

- Call **GetOrderItems** to populate the data.
- Modify the desired order items.
- Pass the results to **UpdateOrderItems**.

The code listing below shows an example of updating an existing item.

Code Listing 76: Updating an Item in an Order

---

```
NameGuid[] itemNames =
    orderService.getOrderItemNamesByOrder(token, orderGuid);
String[] orderItemGuids = new String[] { itemNames[0].getGuid()
};
OrderItem[] items = orderService.getOrderItems(token,
orderItemGuids);
items[0].setQuantityOrdered((short) 22);

orderService.updateOrderItems(token, items, null);
```

---

## Removing Order Items

You may remove items from an order using the **RemoveOrderItems** operation as shown below:

Code Listing 77: Removing an Item from an Order

---

```
itemNames = orderService.getOrderItemNamesByOrder(token,
orderGuid);
orderItemGuids = new String[] { itemNames[0].getGuid() };
orderService.removeOrderItems(token, orderItemGuids);
```

---

## Removing Orders and Canceling Orders

To delete a specific provider order(s), use the **RemoveProviderOrders** operation. *You cannot delete orders that you have already submitted.* For post-submittal orders, a registered user can use the **CancelOrder** operation on the **OrderManagementService** to cancel an open order.

Guest orders cannot be deleted in the post-submittal phase; the **CancelOrder** operation will delete the entire order, including all associated orders.

## Setting User Information for an Order

Every order must have its own user-specific information attached to it so that a data provider can process the order. Contact information, billing and shipping addresses are all required as well as user domain and region. Each order is independent of other orders, so user information must be set up for each order created.

You may set up user information at any point in the order process prior to submission. User information is not required until you validate, quote or submit the order. If the required information is not present at that time, you will receive an error. You can set up user information for an order by calling the **SetUserInformationForOrder** operation on the **OrderManagementService**.

## Validating an Order

To ensure that an order is complete, with all the required options and user-associated information for that order filled out, ECHO validates all orders when submitted or quoted. If you attempt to submit an invalid order, you will receive an error message. Currently, the message reflects only the first validation error encountered; you need to fix the reported error and revalidate until the order is error-free. At this point, if you change any aspect of the validated order before finally submitting it, then the order will again require validation. To validate orders, call **SubmitOrder** or **QuoteOrder**. You can manually validate orders with the **ValidateOrder** operation in the **OrderManagementService**.

Keep in mind that running the **ValidateOrder** operation may return a validation error related to one of the many provider-specific validation rules, which may change over time. You should find the error messages complete enough to help you determine what aspects of the order you need to correct.

Note that validation errors can occur if the order specified does not exist, or does not belong to you.

## Requesting a Quote for an Order

Requesting a quote is an optional step that not all providers support. A quoted order provides a binding price for the order as a whole. The binding quote response from each of the necessary providers may take a day or more to receive. You cannot make changes to an order while it is in the quoting process. If you make changes to the order subsequent to the receipt of a quote, the changes may invalidate the quoted price. You may request a quote from a provider by calling **QuoteOrder**.

## Submitting an Order

Once you have fully validated an order and have made no subsequent changes to the order, submit the order to the system using the **SubmitOrder** operation in the **OrderManagementService**.

Specify your preference for receiving order status information using one of the **NotificationLevel** choices shown in Table 9.

**Table 9: ECHO Order Notification Levels**

| Notification Level | Description   |
|--------------------|---|
| VERBOSE            | The system will e-mail you all provider order state changes and status message updates. (See the next table for more on order state changes.) |
| DETAIL             | The system will e-mail you provider order state changes only.   |
| INFO               | The system will e-mail you when a provider order is closed or canceled.   |
| CRITICAL           | The system will e-mail you when the order fails during submission or is rejected by the provider.   |
| NONE               | The system will not send e-mail.  |

If you do not specify any notification level for a registered user's order, ECHO will use your preference value. If you, as a registered user, have not set a preference value, the default value is **INFO**. The notification for all guests' orders is also **INFO**.

Once an order transmits to the system using the **SubmitOrder** operation, ECHO sends each provider order contained within your order to the appropriate provider. Each provider will decide whether to accept your provider order and how to process it.

### Tracking and Canceling Orders

From this point on, you can track the status of an order using the **GetOrders** operation in the **OrderManagementService**. At any point before the order actually ships, you may request cancellation of the order through the **CancelOrder** operation. You can then use the **GetOrders** operation again to track whether you successfully cancelled the order.

When ECHO sends a request to a provider to quote, submit or cancel a particular order, both ECHO and that provider must have the same ID for that order. ECHO identifies the specific provider order by the ProviderOrderGUID (which is the combination of the provider GUID and the order GUID). However, the provider may also have its own ID system for tracking a particular provider order. As a solution, the provider order in ECHO also contains a provider-tracking ID. When the provider receives the request from ECHO, the provider has the option to accept the order GUID that ECHO has assigned to that provider order, or to provide ECHO with its own unique ID. If the provider does pass back its own unique ID as the provider-tracking ID, then ECHO will use that ID in addition to the normal ECHO ProviderOrderGUID to refer to this provider order. If the provider does not specify a tracking ID, then the provider order will be referenced using only the ECHO ProviderOrderGUID.



## Order States

An order consists of zero or many provider orders. An order state is calculated based on the state of the individual provider orders that make up the order. The table below shows the order states for orders made up of zero or one provider order or of multiple provider orders having the same provider order state.

Table 10: Order States with 0 or 1 Provider Order

| # of Provider Orders | Provider Order State | Order State                  |
|----------------------|----------------------|------------------------------|
| 0                    | -                    | NOT_VALIDATED                |
| 1                    | -                    | Same as provider order state |
|                      | QUOTE_FAILED         | QUOTED_WITH_EXCEPTIONS       |
|                      | QUOTE_REJECTED       |                              |
|                      | SUBMIT_FAILED        | SUBMITTED_WITH_EXCEPTIONS    |
|                      | SUBMIT_REJECTED      |                              |

## Restricted or Deleted Order Items

Some catalog items you may order only with permission. Each provider sets the restrictions and permissions on catalog item ordering. Providers may also delete catalog items at any time. Restrictions can apply to individual users, groups of users or all users. If a catalog item is not available to you, executing the **CreateAndSubmitOrder**, **GetCatalogItemOrderInformation**, **CreateOrder** or **AddOrderItems** operation with this catalog item will return an error message indicating that it is not an orderable item. If an item becomes unavailable after you have created an order, you will receive an error message when you invoke the **UpdateOrderItems**, **ValidateOrder**, **QuoteOrder** or **SubmitOrder** operation on that order.

**This page is intentionally left blank.**

## Chapter 7: Event Notification Service

The Event Notification Service allows you to manage Event Notification Subscriptions and delivery options, allowing ECHO to notify third party applications when an internal event occurs, such as a catalog modification (item added, removed, or updated) or an extended services modification (web service added, removed, or updated). For information on using subscriptions to automate queries, including creating, deleting and deleting subscriptions, refer to “Using Subscriptions to Automate Queries,” Page 36.

The event notification framework supports the delivery of the notifications in various ways, including FTP push, e-mail, and a web service callback. The event notification API is custom to ECHO; however, it follows other WS standards such as WS-Eventing and WS-Notification.

### Event Subscriptions

Event notification subscriptions are represented by **EventSubscriptions**. An **EventSubscription** contains a **GUID**, **Name**, **EndTO** URL, **NotifyTo** URL, **DeliveryMode**, **Expires**. The **Name** must be a unique name for the subscription given by the user. The **EndTO** URL is an optional URL where the notification that the subscription is ending abnormally will be sent. **NotifyTo** URL indicates where all notifications should be sent.

### Delivery Modes

All events are delivered in a format described by the Event Schema. There are three delivery modes for event subscriptions. They are e-mail, FTP push, and HTTPSOAP. E-mail means that as events occur, XML files conforming to the event schema will be sent as e-mail attachments. FTP push will make ECHO upload XML files of events to the FTP server you indicate. HTTPSOAP means that ECHO will make HTTP SOAP web service call to the web service endpoint you give. The web service must implement the Event Sink Service WSDL.

### Filtering Events

Event subscriptions may also optionally have a **Filter**. There are two kinds of **FilterDialects**. The **TOPIC** filter dialect filters events based on the event topic. Every event belongs to a specific topic. Multiple topics may be given by space delimiting them. See the API for the current topics that are supported. The **XPath** filter dialect allows events to be filtered by a boolean XPath applied to the event XML. The event XML will conform to the Event Schema. If the XPath evaluates to true the event notification will be sent.

### Subscription Expiration

All subscriptions have a finite lifetime. An expiration date can be specified when the subscription is created or it will be given a default expiration date if it is not set. Events are NOT sent to the **EndTo** URL if the subscription expires normally.

## Renewing Event Subscriptions

Event subscriptions can be renewed before they expire using the **RenewEventSubscriptions** operation, which will renew an event subscription until the given expiration date. The maximum duration of an event subscription is checked to ensure that the time of the Renew until the expiration date is not exceeded.

## Creating Event Subscriptions

Event subscriptions can be created using the **CreateEventSubscriptions** operation. This allows multiple event subscriptions to be added at once. GUID should be left empty as ECHO will populate this field internally. There are a maximum number of event subscriptions that are allowed to be created for a single user. If the subscription is invalid or the maximum has been reached, the operation will return a fault.

### Code Listing 78: Creating an Event Subscription

---

```
EventNotificationServiceLocator eventNotificationServiceLocator =
    new EventNotificationServiceLocator();
EventNotificationServicePort eventNotificationService =
    eventNotificationServiceLocator.getEventNotificationServicePort();

EventSubscription subscription = new EventSubscription();
subscription.setName("My Event Subscription");

// Notification will be through email
subscription.setDeliveryMode(NotificationDeliveryMode.EMAIL);
subscription.setNotifyTo(new URI("mailto:echotest@gst.com"));

// Filtering on provider and catalog events
subscription.setFilterDialect(FilterDialect.TOPIC);
subscription.setFilter("provider catalog");

String guid =
    eventNotificationService.createEventSubscriptions(userToken,
        new EventSubscription[] { subscription })[0];

System.out.println(subscription.getName() + " has been created");
```

---

## Removing Event Subscriptions

Event subscriptions will eventually expire and be removed on their own, but they can also be removed manually using the **RemoveEventSubscriptions** operation. If an administrator removes the event subscription, then an event notification will be sent to the **EndTo** URL of the subscription if it has been set.

## Chapter 8: Invocation Service

The Invocation Service allows ECHO users to request that ECHO invoke an operation on a registered web service implementation enabling the brokering of services. A user specifies the implementation, operation, and parameters to a service, such as Event Notification Service, and ECHO will invoke the requested service.

ECHO tracks the invocations using an Invocation GUID. Requesting an invocation is an asynchronous operation.

### Limitations

This sections below detail current limitations of the invocation service.

#### Primitive Type Arguments

The Invocation Service can only invoke operations that use primitive types (the basic XML schema primitive types such as “string” and “int”).

#### String Results

ECHO returns the results of an invocation as the type “string.”

#### Invocation State

ECHO stores the state of an invocation using the Status Service and the invocation GUID. The following is a list of the states an invocation can enter:

Table 11: Invocation States

| Invocation State   | Description   |
|--------------------|---|
| ACCEPTED           | The invocation has been accepted by ECHO to be processed.   |
| DELAYED            | The invocation has been delayed by the service partner. A status annotation will list the date the invocation has been delayed until. |
| VALIDATING         | ECHO is currently validating the invocation request.  |
| PREPARING_DATA     | ECHO is preparing data to perform the invocation.   |
| PERFORMING_SERVICE | ECHO is currently invoking the operation.   |

| Invocation State  | Description   |
|-------------------|---|
| PREPARING_RESULTS | The invocation has finished and ECHO is preparing the results of the invocation.                      |
| COMPLETE          | The invocation is complete and the results are available.   |
| ABORTED           | The invocation has been aborted due to error. A status annotation will list the details of the error. |

## Client Operations

This text below lists the operations of the Invocation Service that are specifically intended for end users. Other operations for Service Partners, like **DelayInvocation** and **MarkAsynchronous**, are detailed in the “Invocation Service Aware Web Services” section on the following page. For more information about Service Partners, refer to the forthcoming *ECHO 9.0 Service Partner’s Guide*.

### Invoke Operation

This operation submits a request to invoke an extended service operation. It is an asynchronous method so it returns an Invocation ID, used to track the progress of the invocation.

### Arguments

The operations and parameters on a web service registered in ECHO can be retrieved using the **GetOperations** method on the Extended Services Service. The web service implementation GUID, operation name, and parameters are sent to this operation. The web service implementation GUID must be the GUID of an activated web service implementation registered in ECHO. The operation name is the name of the operation to invoke on the web service. The number and names of the parameters must be correct. The order of the parameters is not important since the names must be unique.

### ECHO Path URI Parameters

If you want to use metadata of granules or collections as parameter values when invoking an operation, then you should use ECHO Path URIs. ECHO Path URIs map to the metadata of a granule or collection. If a parameter value is a URI of the correct format, the URI will be resolved and the value retrieved from the metadata will be sent when invoking the web service.

The type retrieved from the metadata must be convertible to the parameter type in the operation; otherwise, the invocation will be aborted. For example if a parameter takes a float and the ECHO Path URI resolves to “ABZ” then invocation will be aborted. See “Appendix A: ECHO Path URIs” for more information.

## Returned Invocation GUID

The return value is an invocation GUID used to track the invocation. The invocation GUID is passed into all the other methods of the Invocation Service to specify the invocation to which the user is referring. As the invocation is processed, the status is set using the Status Service. The Transaction GUID used in the Status Service is the same as the Invocation GUID.

## Get Results

This returns the last state of the invocation and the results if the invocation has completed. The last status will include the current state of the invocation and any annotations to that state. The results will be included if the invocation state is COMPLETE.

## Service Partner Operations

Service Partners can create web services that are Invocation Service-aware to gain extra abilities and provide more value to end users when being invoked by ECHO. A service called the Invocation Utility Service enables an Invocation Service-aware web service to delay an invocation, add status information on the invocation, or indicate that the web service is asynchronous.

## SOAP Header

Invocation Service-aware web services know they are being invoked by ECHO because a special SOAP Header is passed in the SOAP request to invoke the operation. The information passed is the invocation GUID and the URL to the Invocation Utility Service.

## SOAP Header Layout

This lists the major elements of the ECHO invocation SOAP header. The namespace of all these elements is <http://echo.nasa.gov/echoypes>.

Table 12: Invocation SOAP Header Elements

| Element              | Description   |
|----------------------|---|
| EchoInvocation       | This presence of this tag indicates this is an invocation by ECHO. It holds all of the information. |
| invocationId         | This contains the invocation GUID which will be used on all the Invocation Utility Service methods. |
| invocationUtilityUrl | This contains the URL to the Invocation Utility Service.  |

## SOAP Header Example

This shows an example of what the SOAP header passed during an invocation by ECHO will look like.

### Code Listing 79: Invocation Service SOAP Header Example

---

```
<soapenv:Header>
<ns1:EchoInvocation
  soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
  soapenv:mustUnderstand="0"
  xmlns:ns1="http://echo.nasa.gov/echoTypes">
  <ns1:invocationId
    soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
    soapenv:mustUnderstand="0"
    xsi:type="soapenc:string"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
    88A3BDFB-FDA3-4BA6-F3CE-92D052284503
  </ns1:invocationId>
  <ns1:invocationUtilityUrl
    soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
    soapenv:mustUnderstand="0"
    xsi:type="soapenc:string"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
    http://localhost:28000/echo_v8/InvocationUtilityService
  </ns1:invocationUtilityUrl>
</ns1:EchoInvocation>
</soapenv:Header>
```

---

## Invocation Utility Service

The Invocation Utility Service offers methods to Invocation-aware web services.

### Delay Invocation

This operation delays the invocation until the specified date. This is useful if the web service does not currently have the resources available to process the invocation. ECHO will attempt to invoke the web service again at the specified date and time. See section on timing below.

### Set Status

This operation allows Invocation aware web services to provide detailed information to end users. The status string passed will be stored in ECHO under the current state of the invocation, which can be retrieved and viewed by users using the Invocation Service or Status Service.



### ***Mark Asynchronous***

This operation will mark the web service as being asynchronous. ECHO normally assumes when the operation returns that it is complete but if this operation is called, then the invocation state will stay in `PERFORMING_INVOCATION` until the web service notifies ECHO that it is complete using the Mark Complete operation. See the “Timing Concerns” section below on timing concerns with this operation.

### ***Mark Complete***

This operation is used by asynchronous web services to notify ECHO that they are complete. The string results of the web service operation are an argument to this operation. Web services calling this method should have previously called the Mark Asynchronous operation.

### ***Timing Concerns***

When a web service being invoked by ECHO returns, ECHO normally assumes the invocation is complete and immediately marks it that way. If a web service wants to delay the invocation or mark it asynchronous, it must call the Delay Invocation and Mark Asynchronous operations before returning from the original invocation.

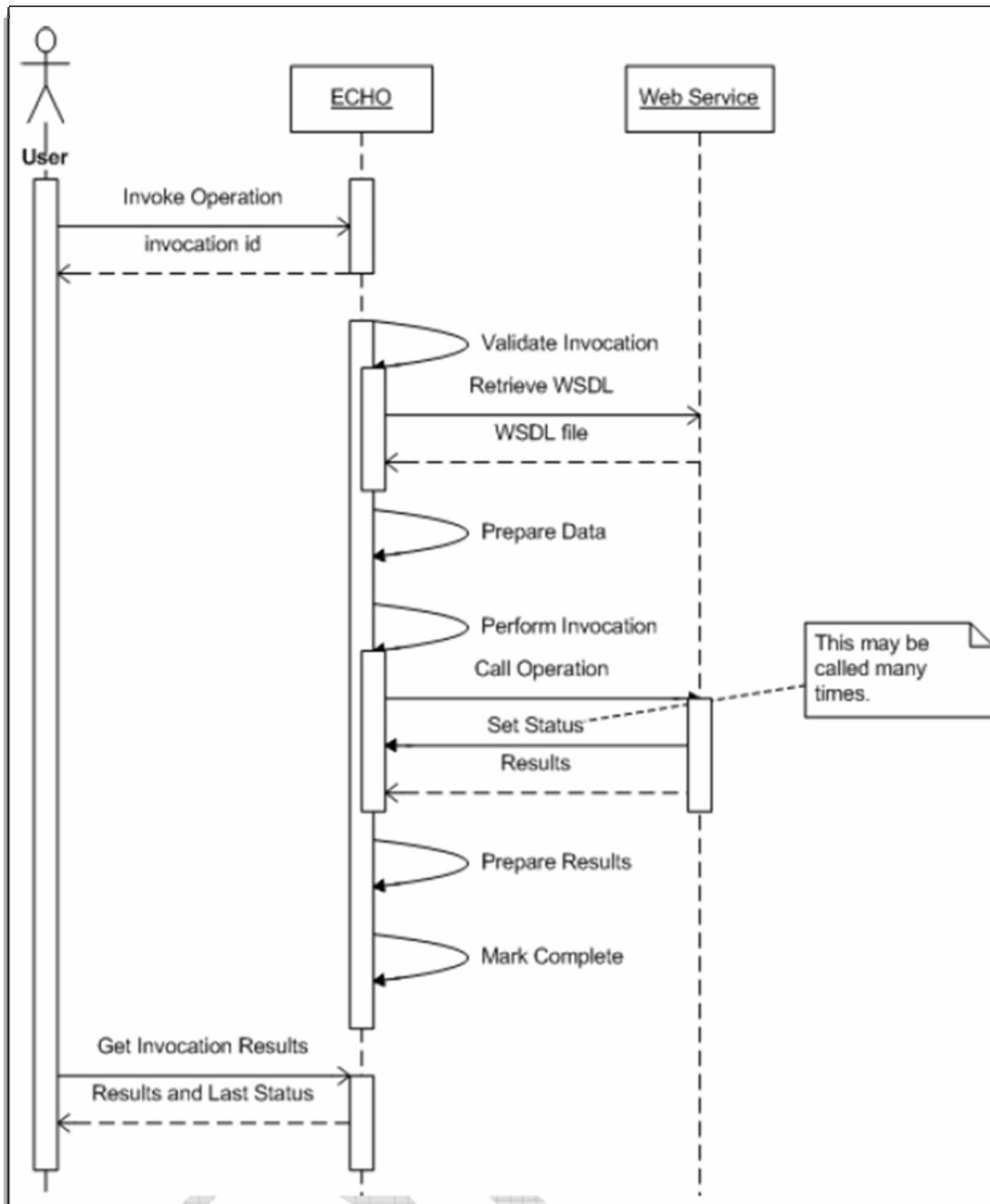
### ***Invocation Sequence Diagrams***

This contains some sequence diagrams showing the steps that happen during a regular and asynchronous invocation.

### Normal Invocation Sequence Diagram

This diagram shows the normal sequence of steps when an invocation occurs.

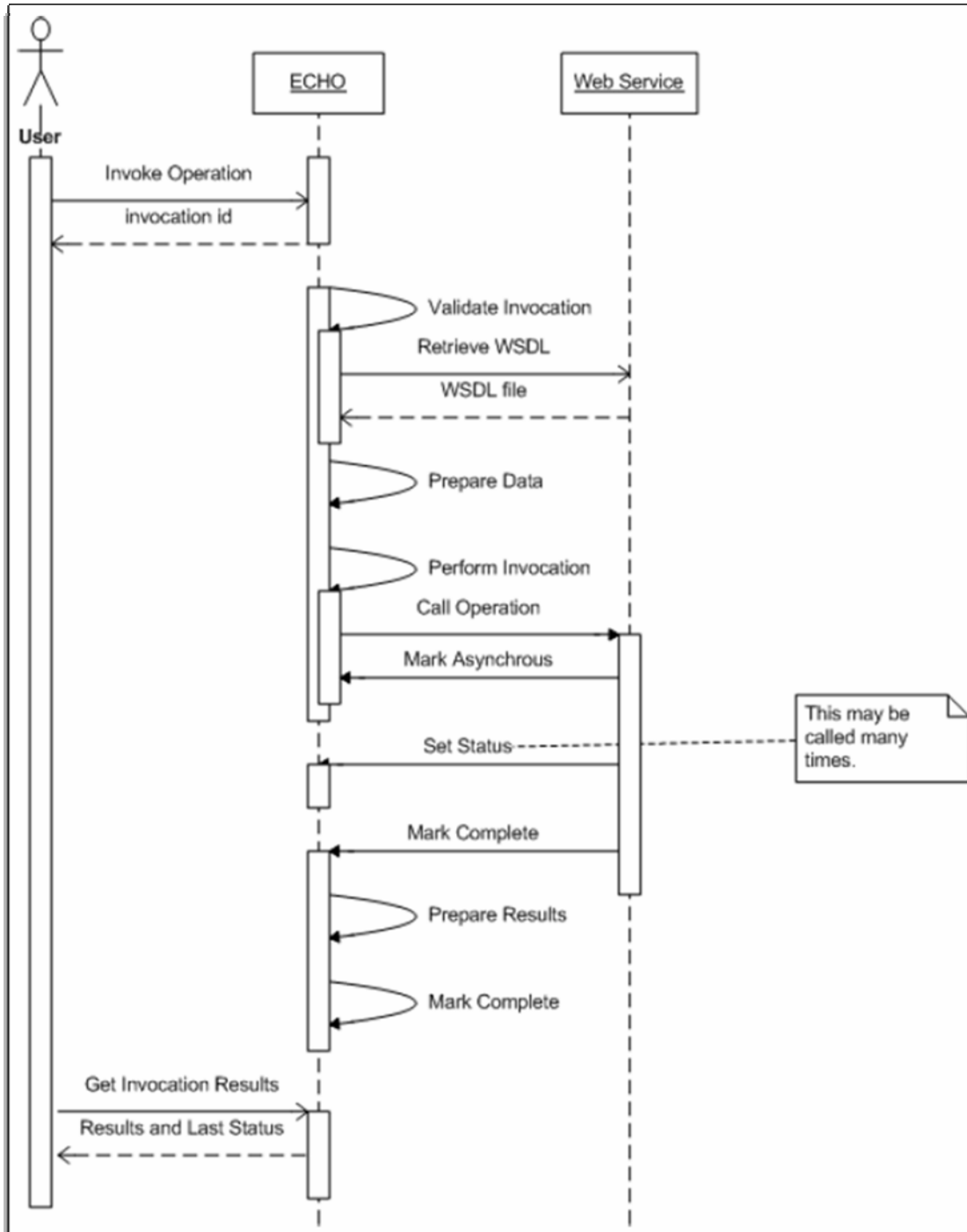
Figure 8: ECHO Invocation Service Sequence



### Asynchronous Invocation Sequence Diagram

This diagram shows the sequence of steps when an invocation occurs on an asynchronous web service.

Figure 9: Asynchronous Invocation Sequence



**This page is intentionally left blank.**

## Appendix A: ECHO Path URIs

ECHO Path URI is the name given to a specific URI which maps into the metadata of a granule or collection.

### Format

The general format for the URI is

`<echoItemType>://<echoHost>/<echoItemId>[/xpath]`

- **echoItemType** := granule | collection This indicates whether the URI is pointing to a granule or collections metadata
- **echoHost** is the address where ECHO server is located. This should be left blank.
- **echoItemId** is the item id of the of the granule or collection
- **xpath** is optional. It is an XPath statement that maps into the XML payload returned from a GetMetadata request using the echoItemId.

Here is an example in the correct format. (Notice the echo host is left blank.)

- collection:///C14016455-PSATEST/%2Fresults%2Fprovider%2Fresult%2FCollectionMetaData%2FECHOItemId%2Ftext%28%29
- The XPath above is escaped to be put in the URI. The un-escaped format looks like: “/results/provider/result/CollectionMetaData/ECHOItemId/text()”

### Behavior

The granule mapping URI will retrieve data from the XML metadata of a granule or collection. The behavior that appears depending upon the state of the XPath selection appears below.

| XPath Selection                                       | Granule   | Collection  |
|---|---|---|
| No XPath included                                     | Returns the entire metadata XML                     | Returns the entire metadata XML                     |
| XPath selects multiple nodes                          | Returns XML fragment representing multiple nodes    | Returns XML fragment representing multiple nodes    |
| XPath selectgs a node with child nodes and attributes | Returns XML fragment represent node and child nodes | Returns XML fragment represent node and child nodes |
| XPath selects a node with a single value              | Returns single value                                | Returns single value                                |

**This page is intentionally left blank.**

## Appendix B: Java-Specific Information

This section discusses one known issue and a solution for that issue.

### Java: Axis Time-out Occurs While Calling Long Running Methods

Sometimes in operations that take longer than five minutes to perform, Axis times out. If this occurs, change the time-out on the service in Axis. In the code example below, the service retrieved from the service location must be cast to a Stub before calling **setTimeout** on it

---

```
FServiceLocator loc = new FServiceLocator();
FServicePort service = loc.getFServicePort();
org.apache.axis.client.Stub s = (Stub) service;
s.setTimeout(timeInMillis); // timeout is in milliseconds
```

---

**This page is intentionally left blank.**



## Appendix C: Acronyms Used in ECHO

You will find the following acronyms frequently used in discussions of ECHO. This list is regularly updated at [http://www.echo.nasa.gov/overview/over\\_acronyms.shtml](http://www.echo.nasa.gov/overview/over_acronyms.shtml).

**ACL** - Access Control List  
**API** - Application Programming Interface  
**AQL** - Alternative Query Language  
**ASF DAAC** - Alaska Satellite Facility DAAC  
**ASTER** - Advanced Spaceborne Thermal Emission and Reflection Radiometer  
**BMGT** - Bulk Metadata Generation Tool  
**COTS** - Commercial Off The Shelf  
**DAAC** - Distributed Active Archive Center  
**DB** - DataBase  
**DTD** - Document Type Definition  
**ECHO** - EOS Clearinghouse  
**ECS** - EOSDIS Core System  
**EDC** - EROS Data Center  
**EDG** - EOS Data Gateway  
**EJB** - Enterprise JAVA Beans  
**EMD** - EOSDIS Maintenance and Development  
**EOS** - Earth Observing System  
**EOSDIS** - EOS Data and Information System  
**EROS** - Earth Resources Observation Systems  
**ESDIS** - Earth Science Data and Information System  
**ESIP** - Earth Science Information Partner  
**ETC** - ECHO Technical Committee  
**FTP** - File Transfer Protocol  
**GCMD** - Global Change Master Directory  
**GES DAAC** - GSFC Earth Sciences DAAC  
**GHRC** - Global Hydrology Resource Center  
**GIS** - Geographic Information System  
**GML** - Geography Markup Language  
**GMT** - Greenwich Mean Time  
**GSFC** - Goddard Space Flight Center  
**GUI** - Graphical User Interface

**GUID** - Globally Unique Identifier  
**IIMS** - Independent Information Management Subsystem  
**J2EE** - Java 2 Enterprise Edition  
**LAADS** - Level 1 and Atmosphere Archive and Distribution System  
**LP DAAC** - Land Processes DAAC  
**MISR** - Multiangle Imaging SpectroRadiometer  
**MODIS** - Moderate Resolution Imaging Spectroradiometer  
**NASA** - National Aeronautics and Space Administration  
**NSIDC DAAC** - National Snow and Ice Data Center DAAC  
**ODL** - Object Description Language  
**OGC** - OpenGIS Consortium  
**ORNL DAAC** - Oak Ridge National Laboratory DAAC  
**PO.DAAC** - Physical Oceanography DAAC  
**PSA** - Product Specific Attribute  
**PUMP** - Provider User Management Program  
**QA** - Quality Assurance  
**SEDAC** - Socioeconomic Data and Applications Center  
**SOAP** - Simple Object Access Protocol  
**SSC** - Stennis Space Center  
**SSL** - Secure Sockets Layer  
**UDDI** - Universal Description, Discovery and Integration  
**UI** - User Interface  
**UR** - Universal Reference  
**URL** - Uniform Resource Locator  
**UTC** - Universal Time, Coordinated (also called GMT/UTC)  
**WIST** - Warehouse Inventory Search Tool  
**WRS** - Worldwide Reference System  
**WSDL** - Web Services Description Language  
**XML** - eXtensible Markup Language  
**XSLT** - eXtensible Style Language Transformation

## Appendix D: ECHO Error Handling

The ECHO 9.0 Web Service API has advanced error-reporting capabilities. There are 12 types of faults reported by ECHO. They are:

**AuthorizationFault** – Reported by ECHO when a user is not authorized to invoke an operation

**DataSizeLimitFault** – Reported by ECHO to indicate that the data size limit has been exceeded

**DuplicateIdFault** – Reported by ECHO to indicate that an entity with the same ID exists in ECHO already

**InternalFault** – Reported by ECHO when an internal error occurs

**InvalidArgumentFault** – Reported to indicate that one or more arguments passed were invalid

**InvalidStateFault** – Reported to indicate that an action by the client would put an object in ECHO in an invalid state

**InvalidURLFault** – Reported to indicate invalid syntax in a URL or an element of the URL that does not exist

**ItemNotFoundFault** – Reported when the client attempts to access one or more objects that do not exist

**ParseFault** – Reported to indicate that some value could not be parsed

**RemovalFault** - Reported to indicate an error that has occurred during the removal of an object from ECHO

**UnsupportedFeatureFault** - Reported to indicate that a feature was selected that is not supported

**ValidationFault** - Reported to indicate that an object in or passed to ECHO is not valid

All the above fault types extend the basic **EchoFault** type. All faults will include an **ErrorCode**, **SystemMessage**, **Timestamp** of when the error occurred and an **ErrorInstanceId**. An **EchoFault** may also have an **OpsMessage**.

Error codes are strings that uniquely identify an error case in ECHO. Some error codes are reused, such as when a required parameter to an operation was not provided. Operations may associate different messages with specific error codes. If Operations has a message configured for an error code, then that message will be returned with the **EchoFault** in the **OpsMessage** element.

In most instances, receiving a fault from ECHO occurs by catching an **EchoFault** and displaying the Ops Message, System Message, and Error Instance ID to the user.

```
try
{
    // Create authentication service
    AuthenticationServiceLocator authServiceLocator =
        new AuthenticationServiceLocator();
    AuthenticationServicePort authenticationService =
        authServiceLocator.getAuthenticationServicePort();

    ClientInformation clientInfo =
        new ClientInformation();
    clientInfo.setClientId("A Client");
    clientInfo.setUserIpAddress("192.168.1.1");

    // Call login with jdoe as username, mypass as password,
    // and client information
    authenticationService.login("jdoe", "mypass",
        clientInfo, null, null);
}
catch (EchoFault e)
{
    // This exception was likely caused by user input.
    String message = "Could not login to ECHO:";

    if (e.getOpsMessage() != null
        && e.getOpsMessage().length() > 0)
    {
        message += "\nOps Message : " + e.getOpsMessage();
    }
    message +=
        "\nMessage: " + e.getSystemMessage()
        + "\nError Instance Id: "
        + e.getErrorInstanceId();
    System.out.println(message);
}
catch (RemoteException e)
{
    // ECHO could not be reached.
    System.out
        .println("Could not communicate with ECHO : "
            + e.toString());
}
catch (ServiceException e)
{
    // An error occurred while creating the service.
    System.out.println("Could not create ECHO Service : "
        + e.toString());
}
```

---

**InternalFaults** capture errors that are internal to ECHO, such as when ECHO cannot talk to the metadata catalog or when an order attempts to transition to an invalid state. There is nothing a client can do to recover from an **InternalFault**, except to report any information provided with the error to ECHO Operations.

---

## Appendix E: Results DTDs

### ECHO Collection Results DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Partly Generated using UniversalRelation.generateDTD
      DTD for ECHO Colletion Results

      $Author$

      Author:  Maitreyee Pasad
      Created: 2/5/2002
      LastUpdated:  $Date$

-->
<!ELEMENT results (provider)*>
<!ELEMENT provider (result)*>
<!ATTLIST provider
      name CDATA #REQUIRED
>
<!ELEMENT result (CollectionMetaData)*>
<!ATTLIST result
      number CDATA #IMPLIED
      itemId CDATA #REQUIRED
      not_visible CDATA "false"
      deleted CDATA "false"
>
<!ELEMENT CollectionMetaData (ECHOItemId, ShortName?, VersionId?,
DataSetId, InsertTime?, LastUpdate?, ECHOInsertDate?, ECHOLastUpdate?,
LongName?, CollectionDescription?, RevisionDate?, SuggestedUsagel?,
ProcessingCenter?, ProcessingLevelId?, ProcessingLevelDescription?,
ArchiveCenter?, VersionDescription?, CitationforExternalPublication?,
CollectionState?, MaintenanceUpdateFrequency?, RestrictionFlag?,
RestrictionComment?, Type?, Price?, TotalGranules?,
SizeMBTotalGranules?, Orderable?, DataFormat?, CatalogItemId?,
Spatial?, Temporal?, Contact*, DisciplineTopicParameters*, Platform*,
AdditionalAttributes*, BrowseProducts?, SpatialKeyword*,
TemporalKeyword*, CSDDTDescription*, CollectionAssociation*, Campaign*,
AlgorithmPackage*, OnlineAccessURLs?, CollectionOnlineResources?,
SpatialInfo?, AssociatedDIFs?)>
<!ELEMENT ECHOItemId (#PCDATA)>
<!ELEMENT ShortName (#PCDATA)>
<!ELEMENT VersionId (#PCDATA)>
<!ELEMENT DataSetId (#PCDATA)>
<!ELEMENT InsertTime (#PCDATA)>
<!ELEMENT LastUpdate (#PCDATA)>
<!ELEMENT ECHOInsertDate (#PCDATA)>
<!ELEMENT ECHOLastUpdate (#PCDATA)>
<!ELEMENT LongName (#PCDATA)>
<!ELEMENT CollectionDescription (#PCDATA)>
<!ELEMENT RevisionDate (#PCDATA)>
<!ELEMENT SuggestedUsagel (#PCDATA)>
<!ELEMENT ProcessingCenter (#PCDATA)>
<!ELEMENT ProcessingLevelId (#PCDATA)>
```

```

<!ELEMENT ProcessingLevelDescription (#PCDATA)>
<!ELEMENT ArchiveCenter (#PCDATA)>
<!ELEMENT VersionDescription (#PCDATA)>
<!ELEMENT CitationforExternalPublication (#PCDATA)>
<!ELEMENT CollectionState (#PCDATA)>
<!ELEMENT MaintenanceUpdateFrequency (#PCDATA)>
<!ELEMENT RestrictionFlag (#PCDATA)>
<!ELEMENT RestrictionComment (#PCDATA)>
<!ELEMENT Type (#PCDATA)>
<!ELEMENT Price (#PCDATA)>
<!ELEMENT TotalGranules (#PCDATA)>
<!ELEMENT SizeMBTotalGranules (#PCDATA)>
<!ELEMENT Orderable (#PCDATA)>
<!ELEMENT DataFormat (#PCDATA)>
<!ELEMENT CatalogItemId (#PCDATA)>
<!ELEMENT Spatial (SpatialCoverageType?, HorizontalSpatialDomain?,
VerticalSpatialDomain*, OrbitParameters?,
GranuleSpatialRepresentation?, GranuleSpatialInheritance?)>
<!ELEMENT SpatialCoverageType (#PCDATA)>
<!ELEMENT HorizontalSpatialDomain ((ZoneIdentifier?, Geometry) |
Global)>
<!ELEMENT ZoneIdentifier (#PCDATA)>
<!ELEMENT Global EMPTY>
<!ELEMENT Geometry (CoordinateSystem, (Point | Circle |
BoundingRectangle | GPolygon | Polygon | Line)+)>
<!ELEMENT CoordinateSystem (Cartesian | Geodetic)>
<!ELEMENT Cartesian EMPTY>
<!ELEMENT Geodetic EMPTY>
<!ELEMENT Point (PointLongitude, PointLatitude)>
<!ELEMENT PointLongitude (#PCDATA)>
<!ELEMENT PointLatitude (#PCDATA)>
<!ELEMENT Circle (CenterLatitude, CenterLongitude, Radius)>
<!ELEMENT CenterLatitude (#PCDATA)>
<!ELEMENT CenterLongitude (#PCDATA)>
<!ELEMENT Radius (#PCDATA)>
<!ELEMENT BoundingRectangle (WestBoundingCoordinate,
NorthBoundingCoordinate, EastBoundingCoordinate,
SouthBoundingCoordinate)>
<!ELEMENT WestBoundingCoordinate (#PCDATA)>
<!ELEMENT NorthBoundingCoordinate (#PCDATA)>
<!ELEMENT EastBoundingCoordinate (#PCDATA)>
<!ELEMENT SouthBoundingCoordinate (#PCDATA)>
<!ELEMENT GPolygon (Boundary, ExclusiveZone?)>
<!ELEMENT ExclusiveZone (Boundary)+>
<!ELEMENT Boundary (Point, Point, Point, Point*)>
<!ELEMENT Polygon (SinglePolygon | MultiPolygon)>
<!ELEMENT SinglePolygon (OutRing, InnerRing?)>
<!ELEMENT OutRing (Boundary)>
<!ELEMENT InnerRing (Boundary)>
<!ELEMENT MultiPolygon (SinglePolygon)+>
<!ELEMENT Line (Point, Point+)>
<!ELEMENT VerticalSpatialDomain (VerticalSpatialDomainType?,
VerticalSpatialDomainValue?)>
<!ELEMENT VerticalSpatialDomainType (#PCDATA)>
<!ELEMENT VerticalSpatialDomainValue (#PCDATA)>
<!ELEMENT OrbitParameters (SwathWidth, Period, InclinationAngle)>

```

```

<!ELEMENT SwathWidth (#PCDATA)>
<!ELEMENT Period (#PCDATA)>
<!ELEMENT InclinationAngle (#PCDATA)>
<!ELEMENT GranuleSpatialRepresentation (Cartesian | Geodetic | Orbit |
NoSpatial)>
<!ELEMENT Orbit EMPTY>
<!ELEMENT NoSpatial EMPTY>
<!ELEMENT GranuleSpatialInheritance (#PCDATA)>
<!ELEMENT Temporal (TimeType?, DateType?, TemporalRangeType?,
PrecisionofSeconds?, EndsatPresentFlag?, RangeDateTime*,
SingleDateTime*, PeriodicDateTime*)>
<!ELEMENT TimeType (#PCDATA)>
<!ELEMENT DateType (#PCDATA)>
<!ELEMENT TemporalRangeType (#PCDATA)>
<!ELEMENT PrecisionofSeconds (#PCDATA)>
<!ELEMENT EndsatPresentFlag (#PCDATA)>
<!ELEMENT RangeDateTime (RangeBeginningDate?, RangeBeginningTime?,
RangeEndingDate?, RangeEndingTime?)>
<!ELEMENT RangeBeginningDate (#PCDATA)>
<!ELEMENT RangeBeginningTime (#PCDATA)>
<!ELEMENT RangeEndingDate (#PCDATA)>
<!ELEMENT RangeEndingTime (#PCDATA)>
<!ELEMENT SingleDateTime (CalendarDate?, TimeOfDay?)>
<!ELEMENT CalendarDate (#PCDATA)>
<!ELEMENT TimeOfDay (#PCDATA)>
<!ELEMENT PeriodicDateTime (PeriodName?, Period1stDate?,
Period1stTime?, PeriodEndDate?, PeriodEndTime?, PeriodDurationUnit?,
PeriodDurationValue?, PeriodCycleDurationUnit?,
PeriodCycleDurationValue?)>
<!ELEMENT PeriodName (#PCDATA)>
<!ELEMENT Period1stDate (#PCDATA)>
<!ELEMENT Period1stTime (#PCDATA)>
<!ELEMENT PeriodEndDate (#PCDATA)>
<!ELEMENT PeriodEndTime (#PCDATA)>
<!ELEMENT PeriodDurationUnit (#PCDATA)>
<!ELEMENT PeriodDurationValue (#PCDATA)>
<!ELEMENT PeriodCycleDurationUnit (#PCDATA)>
<!ELEMENT PeriodCycleDurationValue (#PCDATA)>
<!ELEMENT Contact (Role, HoursOfService?, ContactInstructions?,
ContactOrganizationName?, ContactOrganizationAddress*,
OrganizationTelephone*, OrganizationEmail*, ContactPersons*)>
<!ELEMENT Role (#PCDATA)>
<!ELEMENT HoursOfService (#PCDATA)>
<!ELEMENT ContactInstructions (#PCDATA)>
<!ELEMENT ContactOrganizationName (#PCDATA)>
<!ELEMENT ContactOrganizationAddress (StreetAddress?, City?,
StateProvince?, PostalCode?, Country?)>
<!ELEMENT StreetAddress (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT StateProvince (#PCDATA)>
<!ELEMENT PostalCode (#PCDATA)>
<!ELEMENT Country (#PCDATA)>
<!ELEMENT OrganizationTelephone (TelephoneNumber, TelephoneType?)>
<!ELEMENT TelephoneNumber (#PCDATA)>
<!ELEMENT TelephoneType (#PCDATA)>
<!ELEMENT OrganizationEmail (ElectronicMailAddress?)>

```

```

<!ELEMENT ElectronicMailAddress (#PCDATA)>
<!ELEMENT ContactPersons (ContactFirstName?, ContactMiddleName?,
ContactLastName?, ContactJobPosition)>
<!ELEMENT ContactFirstName (#PCDATA)>
<!ELEMENT ContactMiddleName (#PCDATA)>
<!ELEMENT ContactLastName (#PCDATA)>
<!ELEMENT ContactJobPosition (#PCDATA)>
<!ELEMENT DisciplineTopicParameters (DisciplineKeyword?, TopicKeyword?,
TermKeyword?, VariableKeyword?, ECSPParameterKeyword*)>
<!ELEMENT DisciplineKeyword (#PCDATA)>
<!ELEMENT TopicKeyword (#PCDATA)>
<!ELEMENT TermKeyword (#PCDATA)>
<!ELEMENT VariableKeyword (#PCDATA)>
<!ELEMENT ECSPParameterKeyword (#PCDATA)>
<!ELEMENT Platform (PlatformShortName, PlatformLongName?,
PlatformType?, PlatformCharacteristic*, Instrument*)>
<!ELEMENT PlatformShortName (#PCDATA)>
<!ELEMENT PlatformLongName (#PCDATA)>
<!ELEMENT PlatformType (#PCDATA)>
<!ELEMENT PlatformCharacteristic (PlatformCharacteristicName,
PlatformCharacteristicDescription?, PlatformCharacteristicDataType?,
PlatformCharacteristicUnit?, PlatformCharacteristicValue?)>
<!ELEMENT PlatformCharacteristicName (#PCDATA)>
<!ELEMENT PlatformCharacteristicDescription (#PCDATA)>
<!ELEMENT PlatformCharacteristicDataType (#PCDATA)>
<!ELEMENT PlatformCharacteristicUnit (#PCDATA)>
<!ELEMENT PlatformCharacteristicValue (#PCDATA)>
<!ELEMENT Instrument (InstrumentShortName, InstrumentLongName?,
InstrumentTechnique?, NumberOfSensors?, InstrumentCharacteristic*,
Sensor*, OperationMode*, DisciplineTopicParameters*)>
<!ELEMENT InstrumentShortName (#PCDATA)>
<!ELEMENT InstrumentLongName (#PCDATA)>
<!ELEMENT InstrumentTechnique (#PCDATA)>
<!ELEMENT NumberOfSensors (#PCDATA)>
<!ELEMENT InstrumentCharacteristic (InstrumentCharacteristicName,
InstrumentCharacteristicDescription?,
InstrumentCharacteristicDataType?, InstrumentCharacteristicUnit?,
InstrumentCharacteristicValue?)>
<!ELEMENT InstrumentCharacteristicName (#PCDATA)>
<!ELEMENT InstrumentCharacteristicDescription (#PCDATA)>
<!ELEMENT InstrumentCharacteristicDataType (#PCDATA)>
<!ELEMENT InstrumentCharacteristicUnit (#PCDATA)>
<!ELEMENT InstrumentCharacteristicValue (#PCDATA)>
<!ELEMENT Sensor (SensorShortName, SensorLongName?, SensorTechnique?,
SensorCharacteristic*)>
<!ELEMENT SensorShortName (#PCDATA)>
<!ELEMENT SensorLongName (#PCDATA)>
<!ELEMENT SensorTechnique (#PCDATA)>
<!ELEMENT SensorCharacteristic (SensorCharacteristicName,
SensorCharacteristicDescription?, SensorCharacteristicDataType?,
SensorCharacteristicUnit?, SensorCharacteristicValue?)>
<!ELEMENT SensorCharacteristicName (#PCDATA)>
<!ELEMENT SensorCharacteristicDescription (#PCDATA)>
<!ELEMENT SensorCharacteristicDataType (#PCDATA)>
<!ELEMENT SensorCharacteristicUnit (#PCDATA)>
<!ELEMENT SensorCharacteristicValue (#PCDATA)>

```



```

<!ELEMENT OperationMode (#PCDATA)>
<!ELEMENT StorageMedium (#PCDATA)>
<!ELEMENT AdditionalAttributes (AdditionalAttributeDataType?,
AdditionalAttributeDescription?, AdditionalAttributeName,
MeasurementResolution?, ParameterRangeBegin?, ParameterRangeEnd?,
ParameterUnitsOfMeasure?, ParameterValueAccuracy?,
ValueAccuracyExplanation?, AdditionalAttributeValue?)>
<!ELEMENT AdditionalAttributeDataType (#PCDATA)>
<!ELEMENT AdditionalAttributeDescription (#PCDATA)>
<!ELEMENT AdditionalAttributeName (#PCDATA)>
<!ELEMENT AdditionalAttributeValue (#PCDATA)>
<!ELEMENT MeasurementResolution (#PCDATA)>
<!ELEMENT ParameterRangeBegin (#PCDATA)>
<!ELEMENT ParameterRangeEnd (#PCDATA)>
<!ELEMENT ParameterUnitsOfMeasure (#PCDATA)>
<!ELEMENT ParameterValueAccuracy (#PCDATA)>
<!ELEMENT ValueAccuracyExplanation (#PCDATA)>
<!ELEMENT BrowseProducts (BrowseProduct*)>
<!ELEMENT BrowseProduct (BrowseURL, BrowseDescription?,
BrowseSizeInBytes?, BrowseLastUpdate?, BrowseECHOLastUpdate?)>
<!ELEMENT BrowseURL (#PCDATA)>
<!ELEMENT BrowseDescription (#PCDATA)>
<!ELEMENT BrowseSizeInBytes (#PCDATA)>
<!ELEMENT BrowseLastUpdate (#PCDATA)>
<!ELEMENT BrowseECHOLastUpdate (#PCDATA)>
<!ELEMENT SpatialKeyword (#PCDATA)>
<!ELEMENT TemporalKeyword (#PCDATA)>
<!ELEMENT CSDDTDescription (PrimaryCSDDT, Implementation?, CSDDTComments?,
IndirectReference?)>
<!ELEMENT PrimaryCSDDT (#PCDATA)>
<!ELEMENT Implementation (#PCDATA)>
<!ELEMENT CSDDTComments (#PCDATA)>
<!ELEMENT IndirectReference (#PCDATA)>
<!ELEMENT CollectionAssociation (AssociatedShortName,
AssociatedVersionId?, CollectionType?, CollectionUse1?)>
<!ELEMENT AssociatedShortName (#PCDATA)>
<!ELEMENT AssociatedVersionId (#PCDATA)>
<!ELEMENT CollectionType (#PCDATA)>
<!ELEMENT CollectionUse1 (#PCDATA)>
<!ELEMENT Campaign (CampaignShortName, CampaignLongName?,
CampaignStartDate?, CampaignEndDate?)>
<!ELEMENT CampaignShortName (#PCDATA)>
<!ELEMENT CampaignLongName (#PCDATA)>
<!ELEMENT CampaignStartDate (#PCDATA)>
<!ELEMENT CampaignEndDate (#PCDATA)>
<!ELEMENT AlgorithmPackage (AlgorithmPackageName,
AlgorithmPackageVersion?, AlgorithmPackageDescription?)>
<!ELEMENT AlgorithmPackageName (#PCDATA)>
<!ELEMENT AlgorithmPackageVersion (#PCDATA)>
<!ELEMENT AlgorithmPackageDescription (#PCDATA)>
<!ELEMENT OnlineAccessURLs (OnlineAccessURL*)>
<!ELEMENT OnlineAccessURL (URL, URLDescription?, MimeType?)>
<!ELEMENT URL (#PCDATA)>
<!ELEMENT URLDescription (#PCDATA)>
<!ELEMENT MimeType (#PCDATA)>
<!ELEMENT CollectionOnlineResources (OnlineResource*)>

```

```

<!ELEMENT OnlineResource (OnlineResourceURL,
OnlineResourceDescription?, OnlineResourceType,
OnlineResourceMimeType?)>
<!ELEMENT OnlineResourceURL (#PCDATA)>
<!ELEMENT OnlineResourceDescription (#PCDATA)>
<!ELEMENT OnlineResourceType (#PCDATA)>
<!ELEMENT OnlineResourceMimeType (#PCDATA)>
<!ELEMENT SpatialInfo (SpatialCoverage_Type?, AltitudeDatumName?,
AltitudeDistanceUnits?, AltitudeEncodingMethod?, DepthDatumName?,
DepthDistanceUnits?, DepthEncodingMethod?,
DenominatorofFlatteningRatio?, EllipsoidName?, HorizontalDatumName?,
SemiMajorAxis?, GeographicCoordinateUnits?, LatitudeResolution?,
LongitudeResolution?, LocalCoordinateSystemDesc?,
LocalGeoReferenceInformation?, PlanarCoordinateSystem*,
DepthResolution*, AltitudeResolution*)>
<!ELEMENT SpatialCoverage_Type (#PCDATA)>
<!ELEMENT AltitudeDatumName (#PCDATA)>
<!ELEMENT AltitudeDistanceUnits (#PCDATA)>
<!ELEMENT AltitudeEncodingMethod (#PCDATA)>
<!ELEMENT DepthDatumName (#PCDATA)>
<!ELEMENT DepthDistanceUnits (#PCDATA)>
<!ELEMENT DepthEncodingMethod (#PCDATA)>
<!ELEMENT DenominatorofFlatteningRatio (#PCDATA)>
<!ELEMENT EllipsoidName (#PCDATA)>
<!ELEMENT HorizontalDatumName (#PCDATA)>
<!ELEMENT SemiMajorAxis (#PCDATA)>
<!ELEMENT GeographicCoordinateUnits (#PCDATA)>
<!ELEMENT LatitudeResolution (#PCDATA)>
<!ELEMENT LongitudeResolution (#PCDATA)>
<!ELEMENT LocalCoordinateSystemDesc (#PCDATA)>
<!ELEMENT LocalGeoReferenceInformation (#PCDATA)>
<!ELEMENT PlanarCoordinateSystem (PlanarCoordinateSystemID?,
PlanarCoordinateEncodingMet?, PlanarDistanceUnits?,
BearingReferenceDirection?, BearingReferenceMeridian?,
BearingResolution?, BearingUnits?, DistanceResolution?,
AbscissaResolution?, OrdinateResolution?, MapProjectionName?,
MapProjectionPointer?, LocalPlanarCoordinateSystem?,
LocalPlanarGeoReferenceInfo?, GridCoordinateSystemName?)>
<!ELEMENT PlanarCoordinateSystemID (#PCDATA)>
<!ELEMENT PlanarCoordinateEncodingMet (#PCDATA)>
<!ELEMENT PlanarDistanceUnits (#PCDATA)>
<!ELEMENT BearingReferenceDirection (#PCDATA)>
<!ELEMENT BearingReferenceMeridian (#PCDATA)>
<!ELEMENT BearingResolution (#PCDATA)>
<!ELEMENT BearingUnits (#PCDATA)>
<!ELEMENT DistanceResolution (#PCDATA)>
<!ELEMENT AbscissaResolution (#PCDATA)>
<!ELEMENT OrdinateResolution (#PCDATA)>
<!ELEMENT MapProjectionName (#PCDATA)>
<!ELEMENT MapProjectionPointer (#PCDATA)>
<!ELEMENT LocalPlanarCoordinateSystem (#PCDATA)>
<!ELEMENT LocalPlanarGeoReferenceInfo (#PCDATA)>
<!ELEMENT GridCoordinateSystemName (#PCDATA)>
<!ELEMENT DepthResolution (#PCDATA)>
<!ELEMENT AltitudeResolution (#PCDATA)>
<!ELEMENT AssociatedDIFs (DIF+)>

```

```
<!ELEMENT DIF (EntryID)>
<!ELEMENT EntryID (#PCDATA)>
```

## ECHO Granule Results DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Maitreyee
Pasad (Global Science and Technology, Inc.) -->
<!-- Partly Generated using UniversalRelation.generateDTD
      DTD for ECHO Granule Results

      $Author$

      Author:  Maitreyee Pasad
      Created: 2/5/2002
      LastUpdated:  $Date$

-->
<!ELEMENT results (provider)*>
<!ELEMENT provider (result*)>
<!ATTLIST provider
      name CDATA #REQUIRED
>
<!ELEMENT result (GranuleURMetaData)*>
<!ATTLIST result
      number CDATA #IMPLIED
      itemId CDATA #REQUIRED
      not_visible CDATA "false"
      deleted CDATA "false"
>
<!ELEMENT GranuleURMetaData (ECHOItemId, GranuleUR, InsertTime?,
LastUpdate?, DeleteTime?, ECHOInsertDate?, ECHOLastUpdate?,
RestrictionFlag?, RestrictionComment?, Price?, Orderable?, DataFormat?,
CatalogItemId?, CollectionMetaData, DataGranule?, PGEVersionClass?,
RangeDateTime?, SingleDateTime?, SpatialDomainContainer?,
OrbitCalculatedSpatialDomain?, MeasuredParameter?, Platform*,
Campaign*, AdditionalAttributes?, InputGranule?, BrowseProducts?,
OnlineAccessURLs?, GranuleOnlineResources?, TwoDCoordinateSystem?)>
<!ELEMENT ECHOItemId (#PCDATA)>
<!ELEMENT GranuleUR (#PCDATA)>
<!ELEMENT InsertTime (#PCDATA)>
<!ELEMENT LastUpdate (#PCDATA)>
<!ELEMENT DeleteTime (#PCDATA)>
<!ELEMENT ECHOInsertDate (#PCDATA)>
<!ELEMENT ECHOLastUpdate (#PCDATA)>
<!ELEMENT RestrictionFlag (#PCDATA)>
<!ELEMENT RestrictionComment (#PCDATA)>
<!ELEMENT Price (#PCDATA)>
<!ELEMENT Orderable (#PCDATA)>
<!ELEMENT DataFormat (#PCDATA)>
<!ELEMENT CatalogItemId (#PCDATA)>
<!ELEMENT CollectionMetaData (ShortName?, VersionID?, DataSetId,
CollectionAdditionalAttributes?)>
<!ELEMENT ShortName (#PCDATA)>
<!ELEMENT VersionID (#PCDATA)>
```

```

<!ELEMENT DataSetId (#PCDATA)>
<!ELEMENT DataGranule (SizeMBDataGranule?, ReprocessingPlanned?,
ReprocessingActual?, ProducerGranuleID?, DayNightFlag?,
ProductionDateTime?, LocalVersionID?)>
<!ELEMENT SizeMBDataGranule (#PCDATA)>
<!ELEMENT ReprocessingPlanned (#PCDATA)>
<!ELEMENT ReprocessingActual (#PCDATA)>
<!ELEMENT ProducerGranuleID (#PCDATA)>
<!ELEMENT DayNightFlag (#PCDATA)>
<!ELEMENT ProductionDateTime (#PCDATA)>
<!ELEMENT LocalVersionID (#PCDATA)>
<!ELEMENT PGEVersionClass (PGENAME?, PGEVersion)>
<!ELEMENT PGEVersion (#PCDATA)>
<!ELEMENT PGENAME (#PCDATA)>
<!ELEMENT RangeDateTime (RangeEndingTime?, RangeEndingDate?,
RangeBeginningTime?, RangeBeginningDate?)>
<!ELEMENT RangeEndingTime (#PCDATA)>
<!ELEMENT RangeEndingDate (#PCDATA)>
<!ELEMENT RangeBeginningTime (#PCDATA)>
<!ELEMENT RangeBeginningDate (#PCDATA)>
<!ELEMENT SingleDateTime (TimeOfDay?, CalendarDate?)>
<!ELEMENT TimeOfDay (#PCDATA)>
<!ELEMENT CalendarDate (#PCDATA)>
<!ELEMENT SpatialDomainContainer (GranuleLocality*,
VerticalSpatialDomain*, HorizontalSpatialDomainContainer?)>
<!ELEMENT GranuleLocality (LocalityValue?)>
<!ELEMENT LocalityValue (#PCDATA)>
<!ELEMENT VerticalSpatialDomain (VerticalSpatialDomainContainer?)>
<!ELEMENT VerticalSpatialDomainContainer (VerticalSpatialDomainType?,
VerticalSpatialDomainValue?)>
<!ELEMENT VerticalSpatialDomainType (#PCDATA)>
<!ELEMENT VerticalSpatialDomainValue (#PCDATA)>
<!ELEMENT HorizontalSpatialDomainContainer (ZoneIdentifier?, ((Point |
Circle | BoundingBox | GPolygon | Polygon | Line)+ | Orbit |
Global))>
<!ELEMENT ZoneIdentifier (#PCDATA)>
<!ELEMENT Circle (CenterLatitude, CenterLongitude, Radius)>
<!ELEMENT CenterLatitude (#PCDATA)>
<!ELEMENT CenterLongitude (#PCDATA)>
<!ELEMENT Radius (#PCDATA)>
<!ELEMENT BoundingBox (WestBoundingCoordinate,
NorthBoundingCoordinate, EastBoundingCoordinate,
SouthBoundingCoordinate)>
<!ELEMENT WestBoundingCoordinate (#PCDATA)>
<!ELEMENT NorthBoundingCoordinate (#PCDATA)>
<!ELEMENT EastBoundingCoordinate (#PCDATA)>
<!ELEMENT SouthBoundingCoordinate (#PCDATA)>
<!ELEMENT GPolygon (Boundary, ExclusiveZone?)>
<!ELEMENT ExclusiveZone (Boundary)+>
<!ELEMENT Boundary (Point, Point, Point, Point*)>
<!ELEMENT Point (PointLongitude, PointLatitude)>
<!ELEMENT PointLongitude (#PCDATA)>
<!ELEMENT PointLatitude (#PCDATA)>
<!ELEMENT Polygon (SinglePolygon | MultiPolygon)>
<!ELEMENT SinglePolygon (OutRing, InnerRing?)>
<!ELEMENT OutRing (Boundary)>

```

```

<!ELEMENT InnerRing (Boundary)>
<!ELEMENT MultiPolygon (SinglePolygon)+>
<!ELEMENT Line (Point, Point+)>
<!ELEMENT Orbit (AscendingCrossing, StartLat, StartDirection, EndLat,
EndDirection, NumberOfOrbit?)>
<!ELEMENT AscendingCrossing (#PCDATA)>
<!ELEMENT StartLat (#PCDATA)>
<!ELEMENT StartDirection (#PCDATA)>
<!ELEMENT EndLat (#PCDATA)>
<!ELEMENT EndDirection (#PCDATA)>
<!ELEMENT NumberOfOrbit (#PCDATA)>
<!ELEMENT Global EMPTY>
<!ELEMENT OrbitCalculatedSpatialDomain
(OrbitCalculatedSpatialDomainContainer?)>
<!ELEMENT OrbitCalculatedSpatialDomainContainer (OrbitModelName?,
OrbitNumber?, StartOrbitNumber?, StopOrbitNumber?,
EquatorCrossingLongitude?, EquatorCrossingDate?, EquatorCrossingTime?)>
<!ELEMENT OrbitModelName (#PCDATA)>
<!ELEMENT OrbitNumber (#PCDATA)>
<!ELEMENT StartOrbitNumber (#PCDATA)>
<!ELEMENT StopOrbitNumber (#PCDATA)>
<!ELEMENT EquatorCrossingLongitude (#PCDATA)>
<!ELEMENT EquatorCrossingDate (#PCDATA)>
<!ELEMENT EquatorCrossingTime (#PCDATA)>
<!ELEMENT MeasuredParameter (MeasuredParameterContainer*)>
<!ELEMENT MeasuredParameterContainer (ParameterName, QAStats?,
QAFlags?)>
<!ELEMENT ParameterName (#PCDATA)>
<!ELEMENT QAStats (QAPercentMissingData?, QAPercentOutOfBoundsData?,
QAPercentInterpolatedData?, QAPercentCloudCover?)>
<!ELEMENT QAPercentMissingData (#PCDATA)>
<!ELEMENT QAPercentOutOfBoundsData (#PCDATA)>
<!ELEMENT QAPercentInterpolatedData (#PCDATA)>
<!ELEMENT QAPercentCloudCover (#PCDATA)>
<!ELEMENT QAFlags (AutomaticQualityFlag?,
AutomaticQualityFlagExplanation?, OperationalQualityFlag?,
OperationalQualityFlagExplanation?, ScienceQualityFlag?,
ScienceQualityFlagExplanation?)>
<!ELEMENT AutomaticQualityFlag (#PCDATA)>
<!ELEMENT AutomaticQualityFlagExplanation (#PCDATA)>
<!ELEMENT OperationalQualityFlag (#PCDATA)>
<!ELEMENT OperationalQualityFlagExplanation (#PCDATA)>
<!ELEMENT ScienceQualityFlag (#PCDATA)>
<!ELEMENT ScienceQualityFlagExplanation (#PCDATA)>
<!ELEMENT Platform (PlatformShortName, Instrument*)>
<!ELEMENT PlatformShortName (#PCDATA)>
<!ELEMENT Instrument (InstrumentShortName, OperationMode?,
InstrumentCharacteristic*, Sensor*)>
<!ELEMENT InstrumentShortName (#PCDATA)>
<!ELEMENT OperationMode (#PCDATA)>
<!ELEMENT InstrumentCharacteristic (InstrumentCharacteristicName,
InstrumentCharacteristicValue?)>
<!ELEMENT InstrumentCharacteristicName (#PCDATA)>
<!ELEMENT InstrumentCharacteristicValue (#PCDATA)>
<!ELEMENT Sensor (SensorShortName, SensorCharacteristics*)>
<!ELEMENT SensorShortName (#PCDATA)>

```

```

<!--ELEMENT SensorCharacteristics (SensorCharacteristicName,
SensorCharacteristicValue?)>
<!--ELEMENT SensorCharacteristicName (#PCDATA)>
<!--ELEMENT SensorCharacteristicValue (#PCDATA)>
<!--ELEMENT Campaign (CampaignShortName)>
<!--ELEMENT CampaignShortName (#PCDATA)>
<!--ELEMENT AdditionalAttributes (AdditionalAttribute*)>
<!--ELEMENT CollectionAdditionalAttributes
(CollectionAdditionalAttribute*)>
<!--ELEMENT AdditionalAttribute (AdditionalAttributeName,
AdditionalAttributeValue*)>
<!--ELEMENT CollectionAdditionalAttribute
(CollectionAdditionalAttributeName,
CollectionAdditionalAttributeValue*)>
<!--ELEMENT AdditionalAttributeName (#PCDATA)>
<!--ELEMENT AdditionalAttributeValue (#PCDATA)>
<!--ELEMENT CollectionAdditionalAttributeName (#PCDATA)>
<!--ELEMENT CollectionAdditionalAttributeValue (#PCDATA)>
<!--ELEMENT TwoDCoordinateSystem (StartCoordinate1, EndCoordinate1?,
StartCoordinate2, EndCoordinate2?, TwoDCoordinateSystemName)>
<!--ELEMENT StartCoordinate1 (#PCDATA)>
<!--ELEMENT EndCoordinate1 (#PCDATA)>
<!--ELEMENT StartCoordinate2 (#PCDATA)>
<!--ELEMENT EndCoordinate2 (#PCDATA)>
<!--ELEMENT TwoDCoordinateSystemName (#PCDATA)>
<!--ELEMENT InputGranule (InputPointer*)>
<!--ELEMENT InputPointer (#PCDATA)>
<!--ELEMENT BrowseProducts (BrowseProduct*)>
<!--ELEMENT BrowseProduct (BrowseURL, BrowseDescription?,
BrowseSizeInBytes?, BrowseLastUpdate?, BrowseECHOLastUpdate?)>
<!--ELEMENT BrowseURL (#PCDATA)>
<!--ELEMENT BrowseDescription (#PCDATA)>
<!--ELEMENT BrowseSizeInBytes (#PCDATA)>
<!--ELEMENT BrowseLastUpdate (#PCDATA)>
<!--ELEMENT BrowseECHOLastUpdate (#PCDATA)>
<!--ELEMENT OnlineAccessURLs (OnlineAccessURL*)>
<!--ELEMENT OnlineAccessURL (URL, URLDescription?, MimeType?)>
<!--ELEMENT URL (#PCDATA)>
<!--ELEMENT URLDescription (#PCDATA)>
<!--ELEMENT MimeType (#PCDATA)>
<!--ELEMENT GranuleOnlineResources (OnlineResource*)>
<!--ELEMENT OnlineResource (OnlineResourceURL,
OnlineResourceDescription?, OnlineResourceType,
OnlineResourceMimeType?)>
<!--ELEMENT OnlineResourceURL (#PCDATA)>
<!--ELEMENT OnlineResourceDescription (#PCDATA)>
<!--ELEMENT OnlineResourceType (#PCDATA)>
<!--ELEMENT OnlineResourceMimeType (#PCDATA)>

```

## Appendix F: Best Practices for Queries

The following tips and other recommended practices are in outline form rather than paragraphs for quick reading.

### Best Practices for Running Faster Queries

- Limiting end user choices will lead the user down a logical path. This will promote efficiency by preventing the user from having to make as many choices.
- Search for collections first and limit the collection search by data center, also limit spatially or temporally. Limiting the collection will result in a narrower search and a smaller, more focused result set.

### Efficient Spatial Queries

- If you are querying a single Data Partner, name the Data Partner in the query.
- If you are querying a single collection, include the name of the collection in the query as well.
- Queries for smaller spatial regions return faster results than queries for broader regions.
- Queries for spatial regions with fewer points return faster results than queries with more points.

### Notes about Queries

- Queries that return large result sets take a long time to complete, so try to limit the number of items returned.
- Use HITS and not RESULTS when querying.
- Use TupleType to return only the necessary attributes when presenting data. ECHO defaults to all attributes, which will take longer to return and display.
- Return only what the user will see in the first present.
- Pregather the next page of results while the user is examining the current page.

**This page is intentionally left blank.**



# Index

## A

API  
    website location, 9  
AQL  
    defined, 47

## C

Catalog  
    introduction to the term, 5  
Catalog items, 13  
Collection  
    introduction to the term, 5  
Collection Results DTD, 109

## D

Data search, 47  
Discovery, 47  
DTD  
    collection results, 109  
    granule results, 115  
    location for all ECHO DTDs, 4  
    separate DTDs for updating granules, collections, 5  
    website location, 4

## E

ECHO  
    9.0—impetus for creating this version, 8  
    concept, 3  
    design, 3  
    graphical representation of concept, 3  
    impetus for creating 9.0 version, 8  
    reasons for creating 9.0 version, 8  
Entities, 11  
Error handling  
    types of errors, 107

## G

Granule  
    introduction to the term, 5  
Granule Results DTD, 115  
GUID, 11

## L

Logging in, 17

## O

order broker, 7  
Orders  
    adding order items, 85  
    canceling, 86  
    creating, 84  
    introductory discussion, 13  
    removing, 86  
    requesting a quote, 87  
    submitting an order, 87  
    tracking, 88  
    updating, 86  
    validating, 87

## Q

Queries, 23  
    introductory discussion, 12  
    samples, 32  
Query results, 25

## R

Results, 13  
    large sets, 29  
    saving, 31

## S

Service registry, 6  
Subscriptions, 36  
    deleting, 40  
    event, 91  
    listing, 40

## T

Token, 17

## U

User accounts, 19